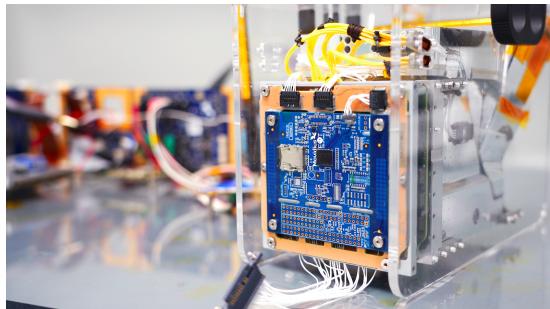


# Predavanje 5

## 4.1 Aktuelnosti

**PhiSat-1** je računar poslan u oktobru 2020. godine u Svetmir. Nalazi se na satelitu i nadgleda površinu planete Zemlje. Ima mogućnost da pravi **multispektralnu analizu** informacija koje dolaze sa optičkog senzora preko **hiperspektralne termalne kamere**. Kamera ima mogućnost da snima u puno više nivoa od optičkih kamera, a pored toga može da snima i različite frekvencije. Na početku su se podaci isključivo snimali i direktno slali na površinu Zemlje. Danas, ovaj model ima mogućnost da snimi i analizira neku sliku, te nakon toga šalje sliku na površinu zemlje. Ukoliko tokom analize nema značajnog sadržaja, slika se jednostavno odbaci. Jedna od primjena je da se snima površina Zemlje zbog požara. U ovom slučaju, često se dešava da je površina prekrivena oblacima, te na taj način dolazimo do zaključka da se slika treba odbaciti. Radi se o prvom čipu koji je odletio u svemir, a isti je izradio **Intel**. Cipovi koje danas imamo na računarima nisu korisni jer bi ih radijacija jako brzo uništila.



Slika 4.1: PhiSat-1

Senzori napreduju dosta brže nego tehnologija koja se koristi za prenos podataka. Stoga, često imamo slike velikih rezolucija čije je slanje jako skupo.

Iz ovog razloga, potrebne su velike količine memorije kako bi se slike spasile na jednom mjestu dok ne dođe vrijeme za njihovo slanje.

## 4.2 Motivacija za kreiranje vještačkih neuronskih mreža

Motivacija za kreiranje vještačkih neuronskih mreža dolazi od studiranja ljudskog mozga. Mozak računa na sasvim drugačiji način od konvencionalnih digitalnih računara. Imamo neke interesantne karakteristike koje možemo iskoristiti kod digitalnih računara kako bismo izvršili modeliranje bazirano na načinu rada mozga. Postoje prednosti modeliranja neurona u odnosu na digitalne računare i suprotno. Recimo, jedna od negativnih karakteristika neurona je što su neuroni pet-šest redova veličine sporiji od digitalne logike ( $ms$  i  $ns$ ). Međutim, ovaj nedostatak u brzini mozak nadoknađuje ogromnim brojem neurona. Istraživanja kažu da ljudski mozak ima oko 100 milijardi neurona i oko  $10^{15}$  konekcija među njima.

Još jedna prednost ljudskog mozga u odnosu na digitalni računar je činjenica da je mozak izuzetno energetski efikasan. Digitalni računar zahtijeva negdje oko  $10^{16} J$  po operaciji u sekundi, dok mozak zahtijeva oko  $10^{-6} J$  po operaciji u sekundi. Sve ove činjenice su bile motivacija za izgradnju vještačkih neuronskih mreža. Mozak je veoma kompleksan, nelinearan sistem i možemo reći da distribuirano procesira informacije na paralelan način. U računarstvu se javlja nova aktuelna oblast koja se naziva **In-memory-computing**. Oblast nalazi motivaciju u radu ljudskog mozga i proučava sistem koji ima mogućnost da izvršava određene operacije i istovremeno snima operacije u jednoj procesnoj jedinici. Današnji računari imaju potpuno odvojena dva koncepta, a to su procesiranje i snimanje podataka. Iz ovog razloga vrlo često imamo *usko grlo* koje nam onemogućava brzo dobavljanje velikih količina podataka za procesiranje.

Postoji projekat **Human Connectome** (HCP) koji služi da bi se izgradile mape kompletnih strukturnih i funkcionalnih neuronskih veza *in vivo* unutar i između pojedinaca. HCP predstavlja prvi veliki pokušaj prikupljanja i razmjene podataka opsega i detalja dovoljnih za započinjanje procesa rješavanja temeljnih pitanja o ljudskog vezivnoj anatomiji i razlikama. Projekat je interesantan zbog činjenice da se izučavanje mozga vrlo često izvršava na materiji koja nije živa. Dakle, radi se o vrlo zahtjevnom poslu koji analizira rad mozga uživo.

## 4.3 Podjela neuronskih mreža

Kao što je već rečeno, neuronske mreže možemo okvirno podijeliti na **biološke neuronske mreže i vještačke neuronske mreže**.

Biološke (ili prirodne) neuronske mreže su biološki organizmi. Primjeri koje nalazimo u praksi su mozak ljudi ili mozak životinja. Radi se o visokosloženim tvorevinama koje imaju mogućnost paralelnog izvršavanja.

Vještačke neuronske mreže su nastale na osnovu motivacije iz proučavanja bioloških neuronskih mreža. Još uvijek se smatra da se radi o dosta primitivnim imitacijama bioloških mreža. Nedostaci se javljaju unutar samih struktura koje zahtijevaju jako puno energije da bi izvršile slične operacije ljudskog mozga. Vještačke neuronske mreže se danas implementiraju na digitalnim računarima opće namjene ili pomoću specijaliziranih kola (analognih, digitalnih ili hibridnih). Danas postoje velike laboratorije koje izučavaju neuronske mreže isključivo pomoću analognih sistema koristeći specijalizirane uređaje.

## 4.4 Neuronska mreža

**Neuronska mreža** generalno predstavlja skup jednostavnih komponenti koje nazivamo **vještačkim neuronima**. Strukture u koje povezujemo te komponente nazivamo **mrežom**.

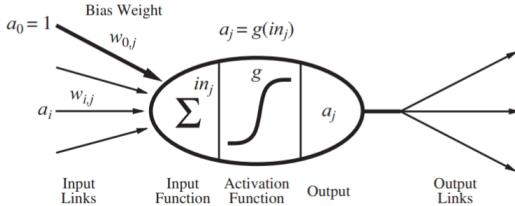
Osobine svake neuronske mreže su utvrđene **topologijom i osobinama pojedinačnih neurona**. Nauka koja se bavi izučavanjem sistema baziranih na osobinama neurona i topologijama neuronskih mreža se naziva **računarska neuronauka** (engl. *computational neuroscience*).

Neuronske mreže imaju mogućnost distribuiranih proračuna. Pored toga, imaju i otpornost na šum kod podataka (**robustnost**). U idealnom slučaju radimo sa podacima koje nemaju šumove jer podaci koji imaju neku vrstu šuma ne predstavljaju dobru reprezentaciju problema kojeg rješavamo. Zbog osobine robustnosti neuronskih mreža, ovakve vrste podataka neće uticati puno na tačnost predikcije neuronske mreže. Najvažnija karakteristika neuronskih mreža jeste sposobnost učenja iz podataka.

Pored neuronskih mreža, postoje i neki drugi sistemi koji imaju mogućnost učenja iz podataka, poput **Bayesove mreže**. Međutim, neuronske mreže postižu dosta bolje rezultate zbog čega su one popularnije.

## 4.5 Model neurona

U nastavku će biti prikazan drugačiji model neurona, iako su koncepti potpuno isti kao i za McCulloch-Pits-ov model.



Slika 4.2: Model neurona

Imamo osnovne funkcionalnosti predstavljene ranijim modelom. Ulazi se sada nazivaju **aktivacijama**, te se označavaju sa  $a_i$ . Svaka aktivacija ima neki koeficijent ili težinu (engl. *weight*)  $w_{i,j}$  koji označava snagu i smjer konekcije (sinapsa). Ova vrijednost se često izražava u opsegu  $[0, 1]$ . Pored toga, koristi se i opseg  $[-1, 1]$  gdje koeficijent može da ima određeni smjer što dalje utiče na rad same mreže. Ovaj koeficijent se podešava prilikom konfiguracije neuronske mreže. Otklon (engl. *bias*) je predstavljen koeficijentom  $a_0$  i predstavlja neku konstantnu vrijednost koja može biti različita od neurona do neurona. Otklon može imati efekat smanjenja ili povećanja uticaja ulaza na aktivacijsku funkciju. Prva funkcija unutar tijela neurona je funkcija sume koja sumira ulazne vrijednosti skalirane sa odgovarajućim koeficijentima. Ova funkcija se većinom označava sa

$$in_j = \sum_{i=0}^n w_{i,j} a_i$$

Vrijednost sume se dalje proslijedi na **aktivacijsku funkciju**  $g$ . Rezultat aktivacijske funkcije predstavlja **izlaz** koji je zadan izrazom

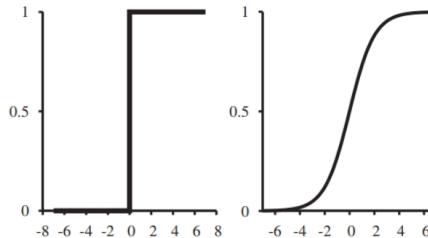
$$a_j = g(\sum_{i=0}^n w_{i,j} a_i)$$

Dakle, izlaz se označava sa  $a_j$  i nosi informaciju dalje kroz sami sistem.

Aktivacijska funkcija osigurava da mreža može predstaviti neku nelinearnu funkciju. Prikazane su dvije najčešće korištene funkcije. Prva funkcija predstavlja **strog prav**. U ovom slučaju, ako su ulazne vrijednosti negativne, izlaz će biti 0. U suprotnom slučaju, dobija se izlaz koji ima vrijednost 1. Ova funkcija ne daje zadovoljavajuće rezultate jer nema puno varijacija preko kojih mreža može naučiti. Još jedan problem ove funkcije predstavlja činjenica da izvod nije definisan za  $x = 0$ .

Drugi primjer predstavlja logističku **sigmoid funkciju** koja ima sasvim drugačije ponašanje i osobine. U ovom konkretnom slučaju, vrijednost

funkcije se definiše u odnosu na sami ulaz, te na izlazu može da primi bilo koju vrijednost koja se nalazi na grafu.



Slika 4.3: Aktivacijske funkcije

Ukoliko imamo strogi prag, onda takav neuron nazivamo **perceptronom**. S druge strane, ako koristimo neku logističku funkciju, onda takav neuron nazivamo **sigmoid perceptronom**.

Nakon što je utvrđen model neurona, potrebno je definisati **topologiju mreže**, odnosno, povezati neurone na odgovarajući način. Postoje dvije osnovne strukture koje se nalaze u praksi koje će se predstaviti u nastavku.

**Mreže bez povratnih veza** (engl. *feed-forward network*) formiraju topologiju unaprijed. Navedeno znači da se veze formiraju samo u jednom smjeru od jednog neurona ka drugom. Stoga, mreža se može predstaviti direktnim acikličnim grafom jer ne postoji povratna sprega. Ne postoji **interni stanje** što znači da se izlaz formira isključivo u odnosu na trenutne ulaze.

**Mreže sa povratnim vezama** (engl. *recurrent network*) predstavljaju mreže dosta popularne u posljednje vrijeme. Kod ovih mreža postoji povratna sprega koja može da dovodi izlaze iz jedne neuronske mreže na ulaz te iste mreže ili na neki njen drugi sloj. Ovakva mreža mora imati mogućnost da pamti i ima neko interni stanje. Obično se radi o dinamičkom sistemu koji može biti u stabilnom stanju ili imati određene oscilacije ili biti u stanju haosa. Stanje sistema ovisi o geneiranom izlazu. Rekurentne mreže se još i nazivaju **sistemima sa kratkom memorijom** (*short-term memory*).

## 4.6 Mreže bez povratnih veza

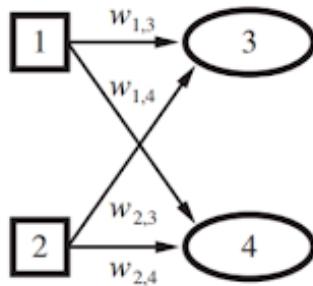
Mreže bez povratnih veza se obično organizuju u **nivoima ili slojevima** (engl. *layers*) na način da jedan neuron može da prihvati ulaze samo ih prethodnog sloja.

Razlikujemo dvije vrste:

1. Naprijed usmjerena jednoslojna mreža (engl. *single layer feed-forward network*)
2. Naprijed usmjerena višeslojna mreža (engl. *multiple layer feed-forward network* ili *Multiple Layer Perceptron (MLP)*)

### Naprijed usmjerena jednoslojna mreža

Naprijed usmjerena jednoslojna mreža se naziva još i **perceptron mreža**. Ulazi su direktno povezani na same izlaze što znači da nemamo skrivenog sloja. Arhitektura se definiše kao broj neurona i način njihovog povezivanja. Sljedećim primjerom je prikazan još jedan način predstavljanja neuronske mreže, gdje 1 i 2 predstavljaju ulaze, a 3 i 4 predstavljaju neurone. Između svake konekcije postoje određeni koeficijenti označeni adekvatnim nazivom ( $w_{1,3}$  predstavlja vezu između ulaza 1 i neurona 3).



Slika 4.4: Perceptron mreža

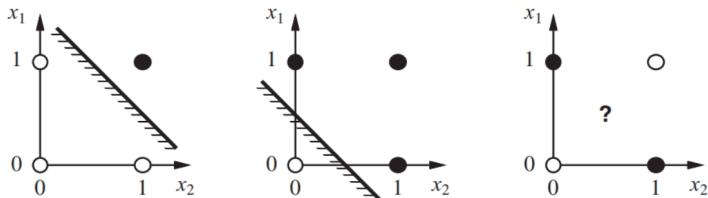
Perceptron mreža sa  $m$  izlaza se može posmatrati kao mreža koja ima  $m$  odvojenih mreža ( $m$  mogućih kombinacija) iz razloga što jedan koeficijent utiče na samo jedan izlaz.

Prikazan je jednostavan primjer iz logičkog dizajna gdje imamo dva ulaza i dva izlaza. Ulazi su označeni sa  $x_1$  i  $x_2$ , a izlazi su označeni sa  $y_3$  i  $y_4$ . Tabelom je prikazana funkcija binarnog sabiranja gdje na izlazu imamo sumu i prenos. Potrebno je odrediti da li pomoću predstavljenog modela neuronske mreže možemo klasificirati funkciju binarnog sabiranja.

$x_1$	$x_2$	$y_3$ (carry)	$y_4$ (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Slika 4.5: Binarno sabiranje

Izlaze čemo predstaviti na grafu zbog lakšeg posmatranja problema.

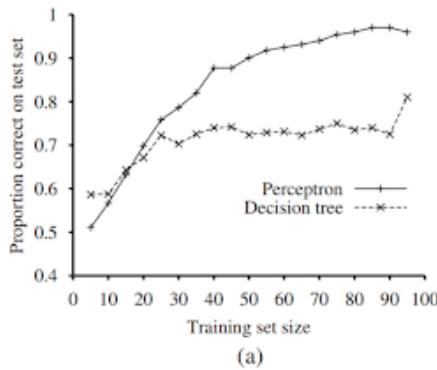


Slika 4.6: Linearna klasifikacija operacija

Prvi graf predstavlja operaciju AND, dok drugi predstavlja operaciju OR. Ove operacije se mogu klasificirati linearnim klasifikatorima koji su označeni ravnim linijama nad datim grafovima. Međutim, za operaciju prikazanu trećim grafom je vizeulno nemoguće pronaći linearnu funkciju koja će razdvojiti ova dva skupa (crne i bijele tačke na grafu).

Ukoliko ponovo pogledamo tabelu binarnog sabiranja na slici 4.5., možemo uočiti da je carry označen sa AND operacijom, a suma odgovara XOR operaciji. Na ovom primjeru uočavamo ograničenja jednoslojnih mreža. Da bismo uspješno mogli klasificirati ovaj problem, potrebno je imati neku nelinearnost što predstavlja složeniji problem. Iako se javlja ovaj nedostatak, to ne znači da su perceptron mreže beskorisne. Ove mreže se mogu koristiti za rješavanje nekih možda čak i težih problema nego što je problem klasificiranja XOR operacije.

Jedan od primjera gdje se ove mreže mogu koristiti je problem izračunavanja **Majority funkcije** koja može da odredi neki izlaz na osnovu vrijednosti većine elemenata. U ovom slučaju, problem ima 11 ulaza gdje je potrebno odrediti izlaz. Izlaz će biti 0 ukoliko većina ulaznih podataka ima neku false vrijednost. U suprotnom slučaju, izlaz će biti 1.

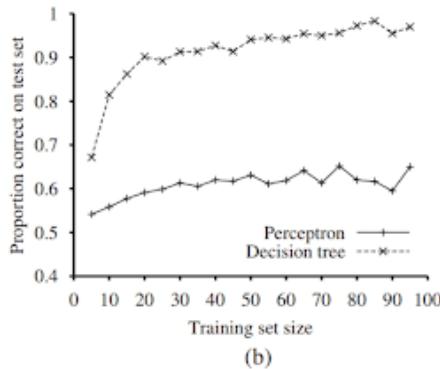


(a)

Slika 4.7: Majority funkcija

Kod ovog problema, na grafu je prikazan broj elemenata nad kojima je vršen trening u odnosu na tačnost samog modela. Dvije linije predstavljene na grafu prikazuju dva moguća rješenja i to uz korištenje **perceptron** i uz korištenje algoritma **stabla odluke** (engl. *decision tree*). Decision tree algoritam predstavlja određeni problem uz pomoć drveta, te se na odnosu određenih parametara donosi neko rješenje. U ovom slučaju, perceptron daje značajno bolje rezultate nego metoda stabla odluke koja je nastala znatno kasnije.

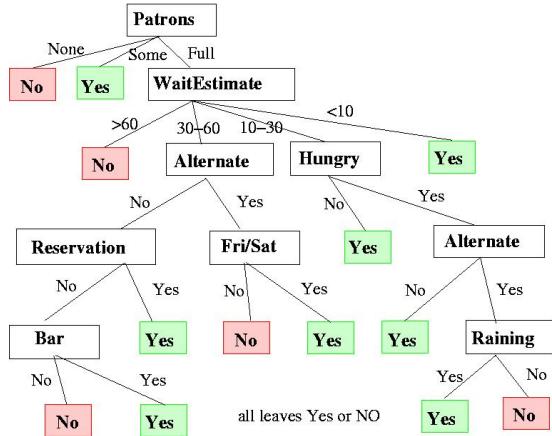
Drugi primjer je problem čekanje na hranu u restoranu. U ovom problemu se javljaju određeni parametri u odnosu na to da li smo gladni, postoji li gužva i slično. Potrebno je donijeti odluku da li se isplati čekati na hranu ili jednostavno otići u neki drugi restoran.



Slika 4.8: Čekanje na hranu u restoranu

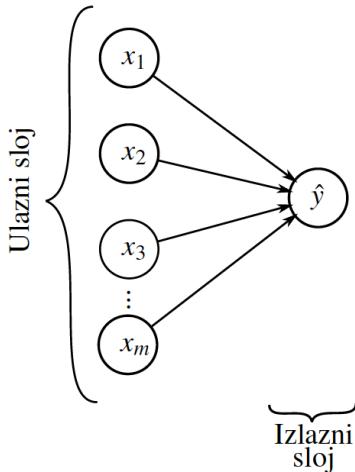
Na ovom primjeru je prikazan algoritam stabla odluke. Na osnovu odgovora na određena pitanja, dolazimo do nekog drugog podpitanja ili dolazimo do finalnog

odgovora.



Slika 4.9: Algoritam stabla odluke

Dakle, naprijed usmjerene jednoslojne mreže predstavljaju jednu od najjednostavnijih mreža. Na sljedećem modelu je prikazan primjer ovakve mreže.

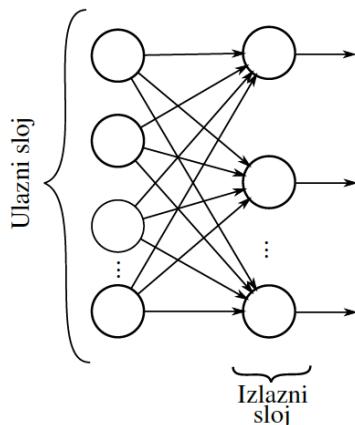


Slika 4.10: Model naprijed usmjerene jednoslojne mreže

Na primjeru imamo nekoliko ulaza predstavljenih sa  $x_i$  i oni predstavljaju atribute koje uzimamo za određeni problem. Recimo da trebamo donijeti odluku da li odobriti ili odbaciti zahtjev za izdavanje pozajmnice. U ovom

slučaju, atributi mogu biti dob aplikanta, godišnja plata, bračni status, posjedovanje imovine i slično. Ovakvih parametara možemo imati proizvoljno mnogo. Numeričke vrijednosti ovih atributa su ulazi u samu mrežu. Kao što je već rečeno, neuroni u ulaznom sloju nemaju mogućnost računanja pa je izlazni sloj jedini sloj unutar ovakvih vrsta neuronskih mreža.

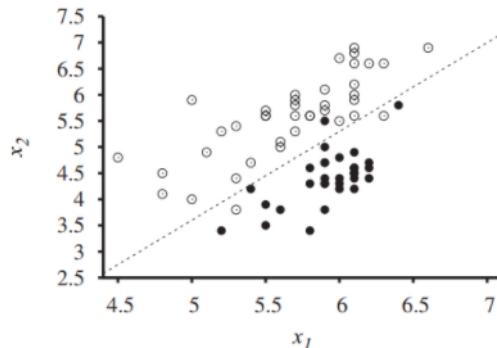
Postoje različite konfiguracije ovakve vrste mreže. Na sljedećem modelu se može vidjeti da za neki broj ulaza možemo imati više različitih izlaza. U primjeru je svaki ulazni sloj je povezan sa svakim neuronom, pa ovaj model nazivamo **potpuno povezanim neuronskim mrežom** (engl. *fully connected neural network*).



Slika 4.11: Potpuno povezana neuronska mreža

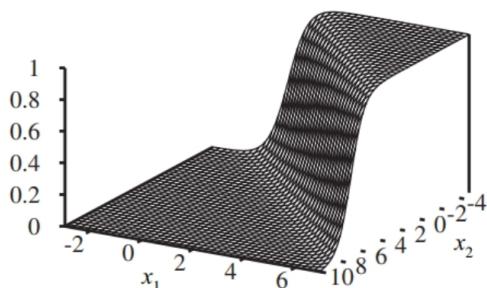
Dakle, kada su neuroni u jednom sloju povezani sa svakim neuronom u sljedećem susjednom sloju, onda za takvu mrežu kažem da je ona potpuno povezana. Pored ovoga, postoje i mreže koje ne moraju biti potpuno povezane i u tom slučaju imamo naizmjenično uključivanje i isključivanje neurona. Ovakve mreže se javljaju kod kompleksnijih konfiguracija.

U nastavku će biti prikazan primjer jednoslojne mreže. Podaci su predstavljeni preko dvije promjenljive  $x_1$  i  $x_2$  koje predstavljaju veličinu tijela i površinu talasa za seizmičke signale. Potrebno je odrediti koje tačke pripadaju klasi zemljotresa (kružići), a koje pripadaju nuklearnoj reakciji (crne tačke). Zadatak klasifikatora je da se nađe funkcija koja će napraviti **granicu** (engl. *decision boundary*) između ova dva skupa. Tražena granica može biti ili linija ili površina (ako imamo više dimenzija).



Slika 4.12: Primjer jednoslojne mreže

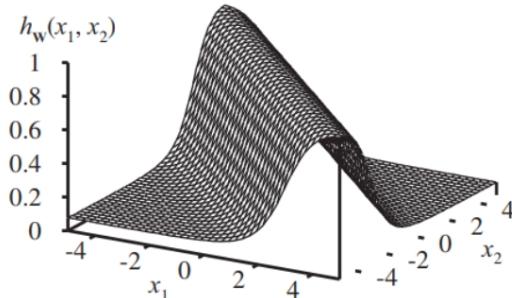
Za rješavanje koristimo neuronsku mrežu. Definišemo aktivacijsku funkciju koja se prikazuje na sljedeći način:



Slika 4.13: Aktivacijska funkcija

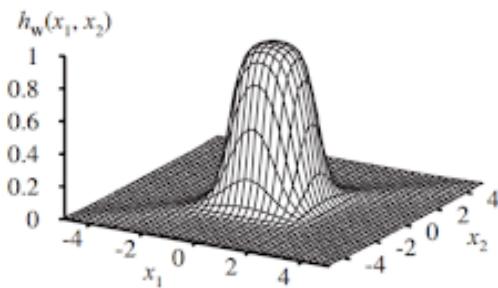
Dakle, imamo dvije dimenzije \$x\_1\$ i \$x\_2\$ i izlaz koji se može kretati od 0 do 1. Ovim je zadana funkcija praga i ima oblik sigmoid funkcije. Pored toga, svaki neuron ima funkciju gubitka što znači da za svaki neuron možemo da generišemo različit broj funkcija gubitka.

Kombinovanjem dvije funkcije praga koje su okrenute suprotno jedna drugoj, možemo dobiti funkciju koju još nazivamo **greben** (engl. *ridge*). U našem primjeru, dobijamo sljedeći greben:



Slika 4.14: Funkcija grebena

Dakle, predstavljena je kombinacijom dvije aktivacijske funkcije i koriste se dva neurona. Ako su izlazi iz ta dva neurona suprotne vrijednosti, možemo generisati datu funkciju grebena. Također, korištenjem više neurona (npr. četiri) sa više aktivacijskih funkcija različitih predznaka možemo doći do funkcije **ispupčenja**.



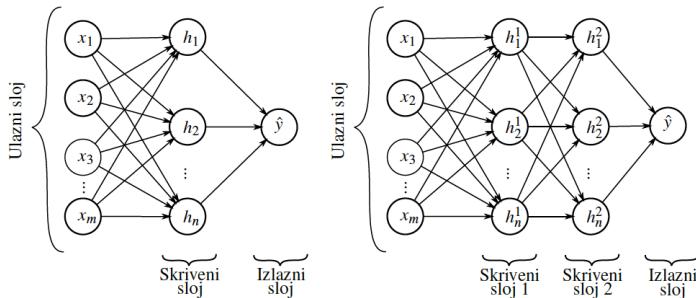
Slika 4.15: Funkcija ispupčenja

Povećavanjem broja neurona u jednom sloju možemo doći do neke proizvoljne funkcije koja će uspješno podijeliti skup podataka na odgovarajuće vrijednosti. Sa jednim skrivenim slojem sa puno neurona moguće je predstaviti bilo koju kontinualnu funkciju, dok je sa više skrivenih slojeva moguće prezentirati i prekide u funkciji.

### Naprijed usmjerena višeslojna mreža

Naprijed usmjerene višeslojne mreže su zasnovane na sličnom konceptu. Dakle, imaju određeni broj ulaza određenim atributima i izlazni sloj.

Razlikuju se po tome što imaju i **skriveni sloj** kojih može biti jako puno. Teoretski nema ograničenja, ali ono se javlja u performansama takve mreže u procesu treniranja. Što imamo veći broj slojeva, imati ćemo kompleksnije proračune čime će izračunavanje tih operacija zahtijevati određeno vrijeme i proces će biti skuplj.



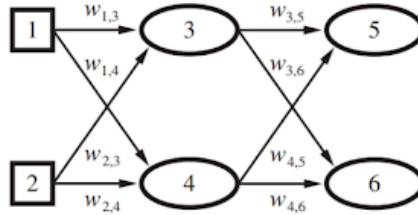
Slika 4.16: Naprijed usmjerena višeslojna mreža

Po broju slojeva, ove mreže možemo podijeliti na **plitke** (engl. *shallow*) i **duboke** (engl. *deep*). Plitke neuralne mreže imaju manji broj skrivenih slojeva, dok duboke imaju veći broj. Svaki skriveni sloj je predstavljen brojem neurona koji se označava sa  $h_n$  s tim što različiti skriveni slojevi ne moraju imati isti broj neurona.

McCulloch i Pitts su 1943. godine predvidjeli ograničenja njihovih modela, te su teoretsali o ideji povezivanja više neurona u složenije mreže. Međutim, u to vrijeme niko nije znao kako trenirati takve strukture. Tek 2012. godine dolazimo u vrijeme kada imamo mogućnost da uspješno treniramo mreže koje imaju 8 skrivenih slojeva i ovom godinom se ulazi u eru dubokog učenja.

Moguće je dati naziv nekoj topologiji ili strukturi na osnovu broja neurona koje se nalaze u ulaznom, skrivenom i izlaznom sloju. Recimo da ulazni sloj ima 10 ulaznih atributa, 6 skrivenih neurona u skrivenom sloju i 3 izlazna neurona, onda se takva mreža naziva 10-6-3 mreža. Dakle, općenito za n ulaznih neurona,  $h_1$  neurona u prvom i  $h_2$  neurona u drugom sloju, takvu mrežu nazivamo  $n-h_1-h_2$  mrežom.

U nastavku je prikazan primjer naprijed usmjerene višeslojne mreže.



Slika 4.17: Model naprijed usmjerenje višeslojne mreže

Mreža je dosta slična jednoslojnoj mreži, s tim što sada postoje još dva neurona 5 i 6. Pravila, što se tiče koeficijenata i veza, su ista kao i u jednoslojnoj mreži. Javljuju se nove veze od neurona 3 i 4 ka neuronima 5 i 6 koje se dobijaju iz aktivacija neurona 3 i 4. Potrebno je izračunati izlaze neurona 5 i 6.

Izlaz iz neurona 5 dobijamo na način da uzmemo otklon  $w_{0,5}$  koji je vezan za neuron 5, te dodamo skaliranu vrijednost aktivacija  $a_3$  i  $a_4$ .

$$a_5 = g(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4)$$

Aktivacije  $a_3$  i  $a_4$  dobijamo na isti način.

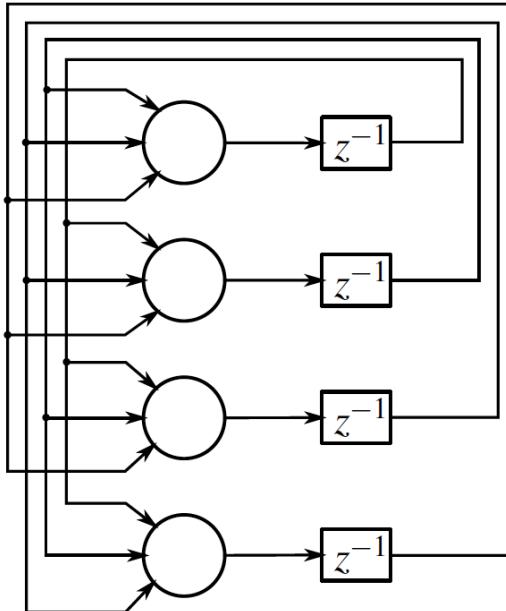
$$a_3 = g(w_{0,3} + w_{1,3}a_1 + w_{2,3}a_2)$$

$$a_4 = g(w_{0,4} + w_{1,4}a_1 + w_{2,4}a_2)$$

Naprijed usmjerna višeslojna mreža nema povratnu spregu što sprječava performanse ove mreže.

## 4.7 Mreže sa povratnim vezama (rekurentne mreže)

Rekurentne mreže se razlikuju po tome što izlaze možemo ponovo dovesti do samih ulaza na istu mrežu, te na ovaj način ponovo trenirati mrežu sa koeficijentima koji su nađeni u međuvremenu. Imaju barem jednu povratnu spregu. Najjednostavniji model je prikazan na slici:

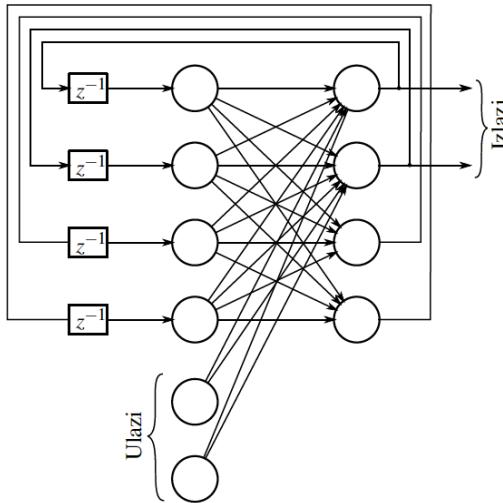


Slika 4.18: Mreža sa povratnim vezama

Na modelu se vidi **kolo zadrške** koje u ovom slučaju pamti samo jedan prethodni element izlaza (da je bilo  $z^{-2}$ , pamtila bi se dva prethodna elementa). Kolo zadrške je predstavljeno  **$z$ -transformacijom**. U ovom primjeru nema skrivenog sloja, a izlazi se dovode na ulaze ulaznih čvorova. Na primjeru imamo četiri izlaza. Međutim, ne dovodimo sva četiri izlaza na svaki ulaz, već samo ona tri različita izlaza u odnosu na dati čvor.

Postojanje povratne sprege u neuronskim mrežama ima značajan uticaj na proces učenja, kao i na performanse mreže. Korištenje povratne sprege uslovljeno je korištenjem kola zadrške koje se označava sa  $z^{-1}$ . Kolo zadrške uvodi jedan trenutak kašnjenja za određeni signal i može uzrokovati nelinearno dinamičko ponašanje mreže.

Sljedećim primjerom je prikazana kompleksnija struktura rekurentne mreže. Ova struktura podrazumijeva jedan skriveni sloj.



Slika 4.19: Mreža sa povratnim vezama

U ovom slučaju, povrat imamo samo kod nekih izlaza i oni se ne dovode na sve ulaze.

## 4.8 Funkcije gubitka

Funkcija gubitka se definiše kao iznos gubitka korisnosti u odnosu na predviđenu vrijednost funkcije  $h(x) = \hat{y}$  kada je tačan odgovor  $f(x) = y$ . Dakle, treba da dâ neku ocjenu razlike između stvarne vrijednosti funkcije koju tražimo  $f(x) = y$  i predikcije koju smo pronašli  $h(x) = \hat{y}$ . Funkcija predikcije predstavlja odabranu hipotezu iz prostora svih hipoteza. Funkcija gubitka se označava sa  $L(x, y, \hat{y})$  ili  $L(y, \hat{y})$ . Pored toga, vrijedi pravilo da funkcija gubitka mora biti jednaka nuli kada računamo ovu vrijednost u odnosu na dvije iste funkcije, odnosno  $L(y, y) = 0$ . U suprotnom, matematički model nije ispravno postavljen.

Recimo da tražena vrijednost funkcije  $y(x)$  za neku određenu tačku daje vrijednost  $y(x) = 137.035999$ . Ako nađemo funkciju predikcije  $h(x)$  koja za određenu tačku daje vrijednost  $h(x) = 137.036$ , postavlja se pitanje da li je funkcija predikcije dovoljno dobra ili ne. U problemima optimizacije neuronskih mreža, isključivo vršimo minimizaciju funkcije gubitka.

Postoji više načina izračunavanja funkcije gubitka. Prvi način je **apsolutna**

**vrijednost gubitka (L1)** data funkcijom

$$L_1(y, \hat{y}) = |y - \hat{y}|$$

dok je drugi način putem **kvadratne vrijednosti greške** definiranom funkcijom

$$L_2(y, \hat{y}) = (y - \hat{y})^2$$

Možemo definisati **generalizirani gubitak** kada su poznati svi ulazno/izlazni primjeri za neki problem, odnosno, potrebno je definisati sve kombinacije ulaza i izlaza. Za veliki broj problema nije moguće definisati ovaj gubitak zbog kompleksnosti.

Neka su ulazno/izlazni primjeri označeni sa  $\varepsilon$ . Generalizirani gubitak se definiše na sljedeći način:

$$\text{GenLoss}_L(h) = \sum_{(x,y) \in \varepsilon} L(y, h(x)) P(x, y)$$

Dakle, generalizirani gubitak se izračunava u odnosu na nađenu hipotezu. Za svaki pojedinačni element u skupu  $\varepsilon$  ćemo izračunati gubitak, te isti pomnožiti sa vjerovatnoćom koja će biti poznata jedino ako su nam poznati svi elementi u posmatranom prostoru. Kada ne znamo sve moguće kombinacije, ovu vjerovatnoću ne možemo izračunati.

Kada nađemo generalizirani gubitak, izračunati ćemo najbolju moguću hipotezu u tom prostoru. Ova hipoteza treba da bude minimalna očekivana vrijednost generaliziranog gubitka:

$$h^* = \operatorname{argmin}_{h \in H} \text{GenLoss}_L(h)$$

gdje je  $H$  skup hipoteza.

Generalizirani gubitak se računa iz razloga što možemo da napravimo jedan model kojem ćemo poslati različite podatke iz istog skupa. Za ove podatke ćemo dobiti različite funkcije od kojih ćemo odrediti koja je najbolja na opisani način.

Ako nije poznata vjerovatnoća distribucije  $P(x, y)$ , moguće je definisati samo **empirijski gubitak** na skupu primjera koji označavamo sa  $E$ :

$$\text{EmpLoss}_{L,E}(h) = \frac{1}{N} \sum_{(x,y) \in E} L(y, h(x))$$

Najbolja estimirana (procjenjena) hipoteza se računa kao minimalna očekivana vrijednost empirijskog gubitka:

$$\hat{h}^* = \operatorname{argmin}_{h \in H} \text{EmpLoss}_{L,E}(h)$$

Postoje četiri razloga zašto će se pronađena funkcija  $\hat{h}^*$  razlikovati od stvarne funkcije  $f$ :

1. **Funkcija f je neostvariva**, odnosno, moguće je da se ne nalazi u prostoru hipoteza ili može da se nalazi, ali da postoje druge funkcije koje su efikasnije.
2. **Funkcija f varira** što znači da algoritam učenja može vratiti drugu funkciju za drugi skup primjera, iako su ti primjeri nastali sa istom funkcijom  $f$ .
3. **Funkcija f je nedeterministička funkcija** ili je pod uticajem šuma, što znači da vrijednost funkcije može biti različita za isti ulaz  $x$ .
4. **Računska kompleksnost** opisuje nerješivost problema ukoliko je prostor hipoteza složen jer u tom slučaju možemo pretraživati samo jedan dio prostora (lokalno pretraživanje).

U našem slučaju, možemo praktično da radimo samo sa empirijskim gubitkom jer se generalizirani gubitak definiše na širi način.

Kada govorimo o učenju iz podataka, može se podijeliti na dva pristupa:

1. **učenje iz malog skupa podataka** - nekoliko desetina do nekoliko hiljada podataka
2. **učenje iz velikog skupa podataka** - milioni podataka

Jasno je da je bolje učiti iz velikog skupa podataka i veže se za **big data** koncept.