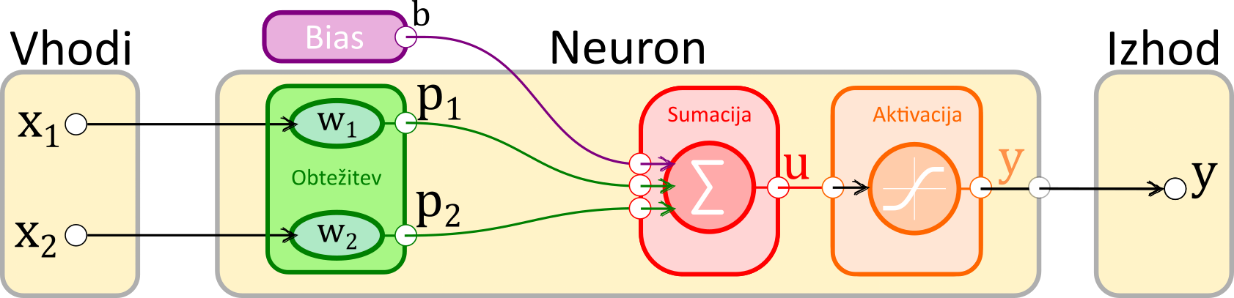
Kako izgradimo nevronsko mrežo

# Predstavitev osnovnih gradnikov

## Nevron

Kot prvo se posvetimo osnovnemu gradniku nevronske mreže, to je nevron. Nevron je v osnovi sestavljen iz voha ali vhodov ter izhoda. Na vhod nevrona vodimo določeno informacijo. Za zdaj si predstavljamo to vhodno informacijo kot vektor, ki ima dve komponenti *x*1 in *x*2. Znotraj nevrona se izvedejo določene matematične operacije, ki kot rezultat proizvedejo izhodno informacijo y.



V notranjosti nevrona se izvedejo tri matematične operacije. V prvem koraku se vsak vhod pomnoži z določeno obtežitvijo. V našem primeru se vhoda x1 in x2 tako pretvorita v:

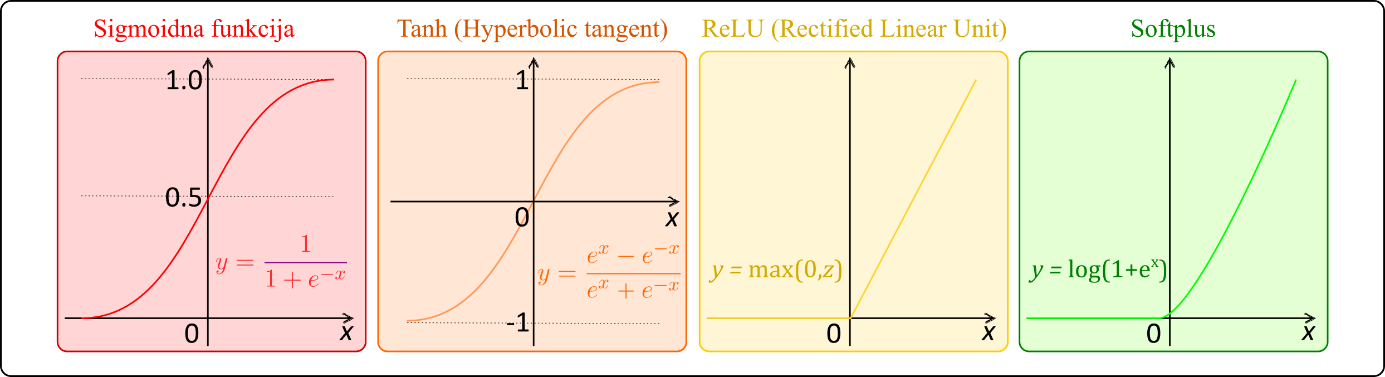
S tako pretvorjenimi vhodnimi podatki, se v nevronu premaknemo naprej do sumacijske točke. V tem koraku nevron sešteje vse signale, ki jih vodimo v sumacijsko točko. V tem koraku dodamo še tako imenovan bias ali offset vrdnost. V slovenščini bi lahko to vrednost poimenovali kot nekakšeno "pristranskost" v smeri pozitivne ali negativne vrednosti, ki se doda obteženemu vhodnemu signalu in tako vpliva na končni izhod nevrona.

V zadnjem koraku vrednost iz sumacijske točke, tj. vrednost , vodimo v aktivacijsko funkcijo nevrona in tako pridobimo vrednost izhoda nevrona oziroma vrednost *y*.

Namen aktivacijske funkcije v nevronu je, da nevronu doda nelinearnost in mu omogoči, da se nauči tudi bolj zapletenih vzorcev v podatkih. **Brez nelinearne aktivacijske funkcije bi namreč izhod nevrona bil linearna funkcija njegovih vhodov in bi bil omejen na reševanje linearnih problemov**. Aktivacijska funkcija prenese obtežene vhodne vrednosti skozi nevronske plasti in jih pretvori v izhodne vrednosti, ki so običajno omejene na določeno območje, na primer med 0 in 1 ali med -1 in 1. Med najpogostejše aktivacijske funkcije spadajo spodaj navedene:

* **Sigmoidna funkcija**: se pogosto uporablja v nevronskih mrežah za binarno klasifikacijo, kjer je cilj določiti, ali je določen primer pripadal enemu ali drugemu razredu. Sigmoidna funkcija pretvori vhodne vrednosti v območje med 0 in 1, kar jo naredi primerno za ta namen.
* **Tanh funkcija**: se pogosto uporablja v nevronskih mrežah, podobno kot sigmoidna funkcija, vendar ima simetrično obliko in pretvori vhodne vrednosti v območje med -1 in 1.
* **ReLU funkcija**: se pogosto uporablja v skritih slojih nevronskih mrež za hitrejše učenje in zmanjšanje verjetnosti, da bo prišlo do problema eksponentno izginjajočega gradienta. ReLU funkcija je definirana kot maksimum med 0 in vhodno vrednostjo.
* **Softplus funkcija**: je nelinearna funkcija, ki se pogosto uporablja v nevronskih mrežah za regresijske probleme. Softplus funkcija je definirana kot logaritem od 1 plus eksponent vhodne vrednosti.

Omenjene aktivacijske funkcije so tudi prikazane v spodnji sliki.



## Primer delovanja nevrona

Imamo nevron, ki ima **dva vhoda** in uporablja **sigmoidno** aktivacijsko funkcijo. Parametri nevrona so:

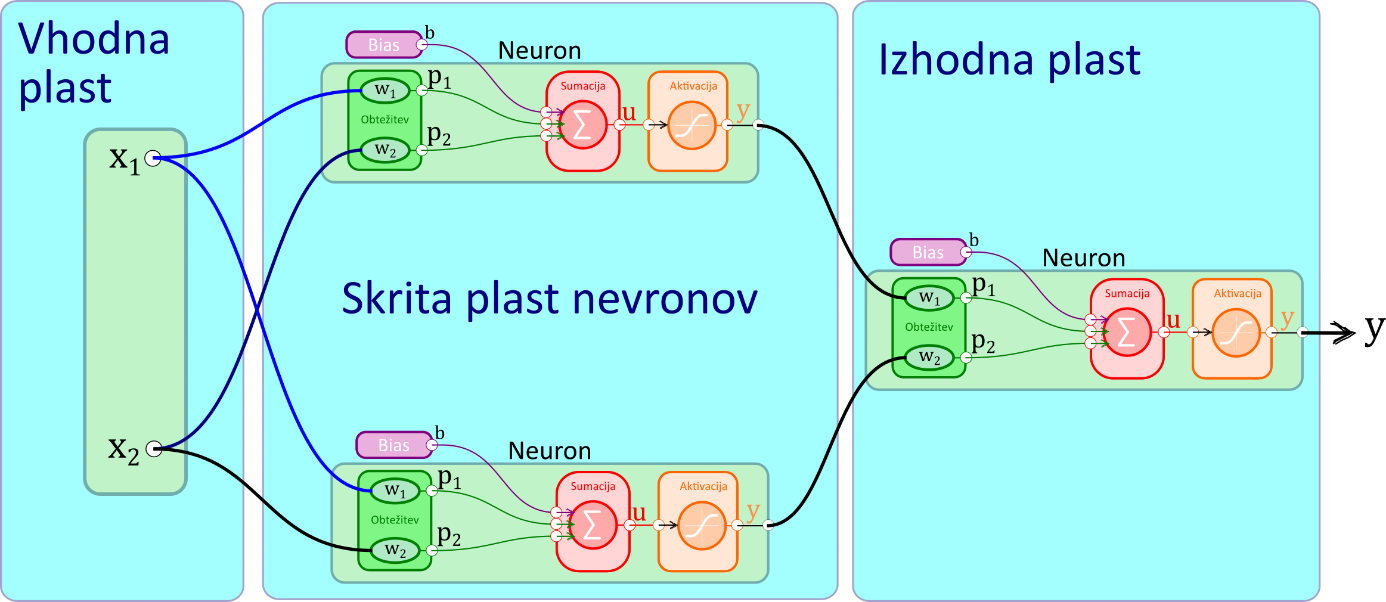
Na vhod nevrona vodimo Za bolj pregledno računanje uporabimo pravila, ki veljajo za vektorsko množenje.

Sedaj lahko vodimo vrednost iz sumacijske točke v aktivacijsko funkcijo nevrona in dobimo njegovo vrednost izhoda.

Tako smo s procesom imenovanim »feedforward« določili izhodno vrednost nevrona.

# Združevanje nevronov v mrežo

V nadaljevanju bomo uporabili večje število nevronov. Nevrone bomo razporedili v **skrito plast nevronov** in **izhodno plast nevronov**. Še zmeraj pa bomo imeli vhodne informacije, ki bodo locirane v **vhodni plastni**. Naša nevronska mreža bo tako sestavljena iz treh plasti vhodne plasti, skrite plasti in izhodne plasti.



Naša nevronska mreža je sestavljena iz dveh vodov, dveh nevronov v skriti plasti in enega nevrona v izhodni plasti. Tukaj naj še omenimo, da se skrita plast nanaša na vse nevrone oziroma plasti nevronov, ki se nahajajo med vhodno in izhodno plastjo. Tako lahko imamo tudi več plasti nevronov v skriti plasti.

Uporabimo zgornje prikazano nevronsko mrežo in predpostavimo, da imajo vsi nevroni enake uteži *w*=[0,1], enak bias *b*=0 in enako sigmoidno aktivacijsko funkcijo. Nevrona v skriti plasti bomo imenovali nevron h1 in h2, nevron v izhodni plasti pa bo nevron o1. Označimo z h1, h2 in o1 izhode nevronov, ki jih predstavljajo. Za vhod ponovno uporabimo .

Kakšen izhodni signal pričakujem v nevronski mreži?

Ker so vsi nevroni enaki je izhod h1 = h2. Tako lahko zapišemo:

|  |
| --- |
|  |

Če vodimo na prej omenjeno nevronsko mrežo na vhod vrednosti 2 in 3, potem pričakujemo na izhodu vrednost 0.7216.

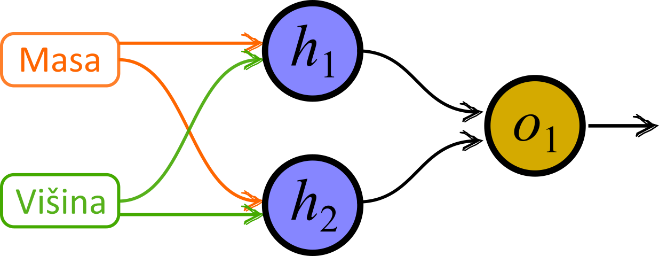
Nevronska mreža lahko ima poljubno število plasti in poljubno število nevronov v teh plasteh. Osnovna ideja ostaja enaka: vhod(i) se prenašajo naprej skozi nevrone v mreži, da se dobijo izhod(i) na koncu. Za preprostost bomo v nadaljevanju tega prispevka še naprej uporabljali zgornje prikazano mrežo.

## Učenje nevronske mreže

Prejeli smo naslednje meritve

| **Ime** | **Masa [kg]** | **Višina [m]** | **Spol** |
| --- | --- | --- | --- |
| Anja | 61 | 1.65 | Ž |
| Blaž | 83 | 1.83 | M |
| Matej | 79 | 1.78 | M |
| Andreja | 54 | 1.53 | Ž |

Našo nevronsko mrežo želimo naučiti, da uporabi podatek o višini in masi in skuša napovedati (klasificirati) vzorec kot moški ali ženski. Poenostavljeno lahko arhitekturo naše nevronske mreže pokažemo kot:



Kot prvo bomo stolpec s podatki o klasifikaciji podatkov, tj. stolpec Spol pretvorili v numeričen zapis, tako bomo M pretvorili v 0 in Ž v 1. Istočasno bomo tudi vrednosti vhodnih parametrov preoblikovali tako, da bomo določili koliko odstopajo od povprečne vrednosti. Povprečna vrednost za stolpec Masa je 62,25 kg in za stolpec Višina 1,6975 m. V spodnji tabeli imamo tako prikazano razliko mase ΔMasa in razliko v višini ΔVišina. Razlika je v obeh primerih določena kot razlika med izmerjeno vrednostjo in povprečno vrendostjo.

| **Ime** | **ΔMasa [kg]** | **ΔVišina [m]** | **Spol** |
| --- | --- | --- | --- |
| Anja | -8.25 | -0.05 | 1 |
| Blaž | 13.75 | 0.13 | 0 |
| Matej | 9.75 | 0.08 | 0 |
| Andreja | -15.25 | -0.17 | 1 |

Da bi mrežo naučili, da zna prepoznati te razrede, glede na vhodne podatke, moramo nastaviti parametre nevronov.

## Izguba (ang. loss)

Izguba ali po angleško imenovana loss je mera, ki nam pove, kako dobro se je nevronska mreža odrezala pri klasifikaciji podatkov. Izgubo določimo s povprečno kvadratno napako (MSE):

V tej enačbi so posamezni parametri;

* *n* – število vzorcev, ki jih klasificiramo
* *y* – se nanaša na spremenljivko, ki jo želimo napovedati
* – so podane vrednosti, ki jih želimo napovedati
* – so vrednosti, ki jih je nevronska mreža napovedala
* – je kvadrat razlike med pravo in napovedano vrednostjo.

Bolj natančne kot so naše napovedi, manjše so vrednosti . Treniranje nevronske mreže je tako povezano s postopkom s katerim zmanjšujemo vrednosti .

### Izračun MSE

Recimo, da naša ne netrenirana nevronska mreža za vse vzorce poda vrednost 0. Kolikšen bi bil v tem primeru MSE.

| **Ime** | **Spol** | **Napovedan spol** |
| --- | --- | --- |
| Anja | 1 | 0 |
| Blaž | 0 | 0 |
| Matej | 0 | 0 |
| Andreja | 1 | 0 |

### Zmanjševanje MSE

Sedaj imamo referenčno natančnost nevronske mreže. Za našo nevronsko mrežo tako dobimo MSE = 0.5. V nadaljevanju želimo izboljšati natančnost naše mreže in tako zmanjšati MSE. Zaradi nadaljnje ponazoritve se osredotočimo na Anjo.

| **Ime** | **ΔMasa [kg]** | **ΔVišina [m]** | **Spol** |
| --- | --- | --- | --- |
| Anja | -8.25 | -0.05 | 1 |

Če se osredotočimo samo na njo, potem je MSE:

Sedaj lahko izgubo, *L*, zapišemo kot funkcijo različnih spremenljivk:

|  |  |
| --- | --- |
|  |  |

Recimo, da bi hoteli nastavljati vrednost . Zanima nas, kako bi se spremenila izguba, če bi spreminjali . V ta namen uporabimo parcialne odvode. Da ta koncept predstavimo, zapišimo člen:

Sedaj pa še moramo dognati, kako izračunati . Spomnimo se, da s *h*1, *h*2 in *o*1 označujemo izhode posameznih nevronov. Tako lahko zapišemo:

Ker vpliva samo na *h*1, lahko v nadaljevanju zapišemo:

Sedaj se še posvetimo aktivacijski funkciji. Če za aktivacijsko funkcijo uporabimo sigmoidno funkcijo lahko zapišemo:

Torej

Optimiziranje sistema na takšen način se imenuje »backpropagation«. Backpropagation ali "backprop" je algoritem za učenje nevronskih mrež z nadzorovanim učenjem. Namen algoritma je prilagoditi uteži in odklone v mreži, da se zmanjša izguba med napovedmi in dejanskimi vrednostmi. Postopek se začne s prvim prehodom skozi mrežo, kjer se vhodni podatki posredujejo skozi nevronske plasti, da se dobi izhodni signal. Izhodni signal se nato primerja s pričakovanimi vrednostmi in izračuna se izguba. Nato se izguba propagira nazaj skozi mrežo, pri čemer se uporablja verižno **pravilo odvoda, da se izračuna gradient izgube glede na uteži in odklone v mreži**. Gradient pove, kako hitro se izguba spreminja glede na majhne spremembe uteži in odklonov.

### Primer izračuna gradient izgube

Pričnimo z optimizacijskim procesom tako, da postavimo uteži v našem sistemu na 1 in bias vrednosti na 0. Za potrebe razlage pa se osredotočimo zgolj na Anjo.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Ime** | **ΔMasa [kg]** | **ΔVišina [m]** | **Spol** | | --- | --- | --- | --- | | Anja | -8.25 | -0.05 | 1 | |

Na izhodu smo dobili vrednost 0,00000006, kar je zelo močno v prid razredu moški (0). Sedaj pa izračunajmo .

Rezultat nam pove, da če bi obtežitev povečali, bi to izzvalo malo povečanje v izgubi, oziroma, bi postala napoved še manj natančna.

### Stohastično učenje nevronske mreže

Sedaj imamo vsa potrebna orodja, da pričnemo učiti nevronsko mrežo. Uporabili bomo optimizacijski algoritem imenovan, stohastični gradientni spust (ang. stochastic gradient descent (SGD)). Algoritem nam bo povedal, kako moramo spremeniti obtežitve v mreži, da zmanjšamo izgubo oz. MSE mreže in posledično povečamo njeno natančnost. Da omenjeno izvedemo moramo uporabiti naslednjo enačbo:

tukaj je konstanta in se imenuje hitrost učenja. Ta konstanta nadzoruje kako hitro se bo mreža učila. Kot vidimo, moramo zgolj odšteti od . Istočasno pa lahko opazimo tudi da:

* pomeni, da se zmanjša, se posledično zmanjša
* pomeni, da se poveča, se posledično zmanjša

Postopek učenja bo zgledal kot:

1. Naključno izberemo enega od vzorcev iz nabora podatkov. To je tudi to, zaradi česar je algoritem stohastičen.
2. Izračunamo parcialne odvode izgube za vse obtežitve in biase.
3. Posodobimo vrednosti obtežitev in biasov v skladu z dobljenimi vrednostmi.
4. Se vrnemo nazaj h koraku 1

V našem primeru privede do:

Slika, ki vsebuje besede diagram

Opis je samodejno ustvarjen

Barva obtežitev ponazarja + (rdeča) ali – (modra) vrednost obtežitve, absolutna vrednost pa je podana z debelino povezave. Za vsak nevron smo zapisali tudi kolikšen vrednost bias-a. V tabeli pa je podano, kolikšen je predviden spol.

Sedaj uporabimo isto nevronsko mrežo, da jo naučimo, kako se naj vede kot:

* Logična IN vrata

Slika, ki vsebuje besede diagram

Opis je samodejno ustvarjen

* Logična ALI vrata

Slika, ki vsebuje besede diagram

Opis je samodejno ustvarjen