Uradi sam – umjetna neuralna mreža

izv. prof. dr. sc. Frane Čačić Kenjerić
Prehrambeno-tehnološki fakultet Osijek

Što je inteligencija?

Na ovo pitanje ne postoji jedinstveni odgovor, međutim može se opisati kao sposobnost mišljenja koje omogućuje snalaženje u prilikama u kojima se ne koristi nagonsko ponašanje kao niti učenjem stečene navike, vještine i znanja [1]. Oko samog ustroja inteligencije postoje neslaganja, dali je inteligencija jedinstvena sposobnost ili je skup širih sposobnosti (numerička, mnemička, perceptivna, spacijalna, …). Ova svojstva živim bićima proizlaze iz centralnog živčanog sustava koji završava u manjem ili većem organu koji nazivamo mozak. Mozak se sastoji od nakupina živčanih stanica, neurona, koji su međusobno povezani tvoreći (neuralnu) mrežu.

Umjetna inteligencija (Artificial Intelligence *eng.*, AI),
pojam koji označava sposobnost računala ili drugih uređaja da ispunjavaju zadatke za koje se smatra da zahtijevaju inteligenciju, također označava i granu računalstva koje se bavi izradom inteligentnih strojeva [2]. Umjetna neuralna mreža predstavlja matematički model, bioloških neuralnih mreža i jeda je od temeljnih oblika koji se koriste za izradu umjetne inteligencije.

[1] inteligencija. *Hrvatska enciklopedija*, *mrežno izdanje.* Leksikografski zavod Miroslav Krleža, 2013. – 2024. Pristupljeno 17.4.2024. <https://www.enciklopedija.hr/clanak/inteligencija>.
[2] artificial intelligence. The American Heritage Dictionary of the English Language, Fifth Edition HarperCollins Publishers, 2022. Pristupljeno 15.4.2024. <https://ahdictionary.com/word/search.html?q=artificial+intelligence>

Kako napraviti umjetnu neuralnu mrežu ?

Odabir programskog jezika za implementaciju modela -> Python
Python je opći programski jezik vrlo popularan zbog svoje jednostavnosti i velikih mogućnosti, više informacija dostupno na www.python.org. Postoji veći broj biblioteka za izradu neuralnih mreža koje omogućuju laganu izradu, međutim ovdje neće biti korištene za izradu modela. Jedina korištena biblioteka jest *Numpy*.
Primjer modela neuralne mreže koja će na temelju tri ulazna svojstva (X1, X2, X3) predvidjeti vrijednost Y (Tablica 1).
Neuralna mreža koristi nadziranu metodu učenja, što znači da mreži dajemo ulaze i željene rezultate tijekom treniranja, a mreža uči kroz podešavanje težinskih faktora s ciljem smanjivanja pogreške.

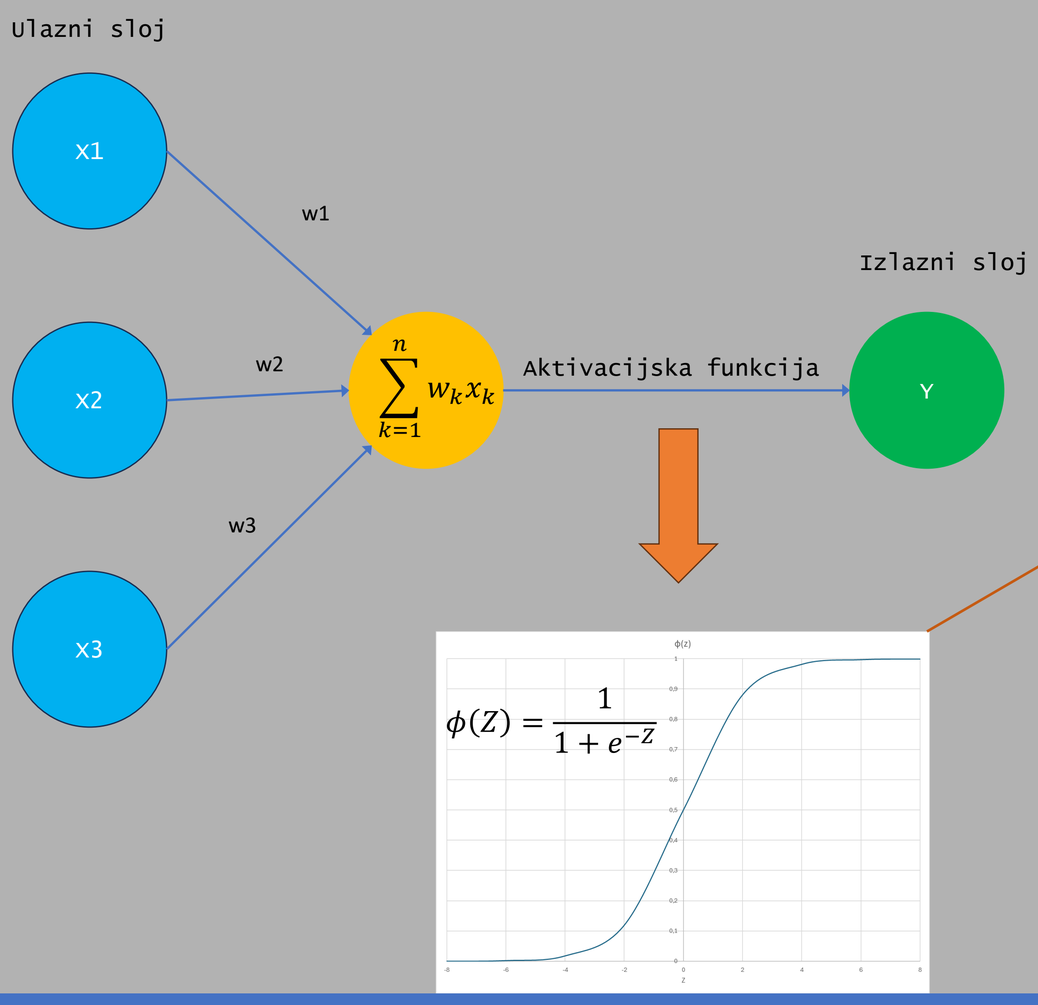
Pogreška->srednje kvadratno odstupanje: $E = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$

Minimiziranje pogreške->algoritam gradijetnog spuštanja: $w_i = w_i - s_u(\frac{\partial E}{\partial w_i})$

$\hat{Y}_i$ *– predviđena vrijednost izlaza,* Y_i *– stvarna vrijednost izlaza,* n *– broj slučajeva,* w_i *– težinski faktor,* s_u *– stopa učenja*

Tablica 1. Podaci za izradu modela umjetne neuralne mreže

Slučaj	X1	X2	X3	Y
A	0	0	0	1
B	0	0	1	0
C	0	1	0	0
D	0	1	1	1
E	1	0	0	1
F	1	0	1	0
G	1	1	0	1
H	1	1	1	0




```
Console 1/A
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
0.007938179928492384
[0.75362104]
[0.24836518]
```

```python
# -*- coding: utf-8 -*-
"""
Spyder Editor
Frane Čačić Kenjerić
Umjetna neuralna mreža
15.04.2024.

"""

import numpy as np


#Ulazne vrijednosti X1,X2,X3 - nezavisne
ulazne_vrijednosti = np.array([[0,0,0],
                               [0,0,1],
                               [0,1,0],
                               [0,1,1],
                               [1,0,0],
                               [1,0,1],
                               [1,1,0],
                               [1,1,1]])

#Izlazne vrijednosti Y - zavisne
izlazne_vrijednosti = np.array([[1,
                                 0,
                                 0,
                                 1,
                                 1,
                                 0,
                                 1,
                                 0,
                                 1,
                                 0]])

izlazne_vrijednosti = izlazne_vrijednosti.reshape(8, 1)  #
transponiranje vektora

#definiranje hiperparametara mreže
np.random.seed(45)
tezinski_faktori = np.random.rand(3,1)
bias = np.random.rand(1)
su = 0.05 # stopa učenja


#definicija funkcije aktivacije - sigmoidna funkcija
def sigmoid(x):
    rez = 1/(1+np.exp(-x))
    return rez

#definicija derivacije funkcije aktivacije
def sigmoid_derivacija(x):
    rez = sigmoid(x)*(1-sigmoid(x))
    return rez

#treniranje modela
n = 30000 # broj epoha, krugova treniranja neuralne mreže

for epoch in range(n):
    # unaprijedna faza
    ulazi = ulazne_vrijednosti
    XW = np.dot(ulazi, tezinski_faktori) + bias # ulaz * tez. faktor
+ bias
    z=sigmoid(XW) # vrijednost neurona
    # propagacija unazad (podešavanje težinskih faktora)
    greska = z - izlazne_vrijednosti
    print(greska.sum())
    dcost = greska
    dpred = sigmoid_derivacija(z)
    z_del = dcost * dpred
    ulazi = ulazne_vrijednosti.T
    tezinski_faktori = tezinski_faktori - su * np.dot(ulazi, z_del)

    for num in z_del:
        bias = bias - su * num


#testiranje mreže s ulaznim vrijednostima [0,0,0]
p_ulaz = np.array([0,0,0])
rezultat = sigmoid(np.dot(p_ulaz, tezinski_faktori) + bias)
# zaokružiti vrijednost na cjelobrojnu 0 ili 1
print(rezultat)

#predikcije mreže [1,1,1]
p_ulaz = np.array([1,1,1])
rezultat = sigmoid(np.dot(p_ulaz, tezinski_faktori) + bias)
print(rezultat)
```

fradic@ptfos.hr