Neuronske mreže

Višeslojni perceptron, *Backpropagation*

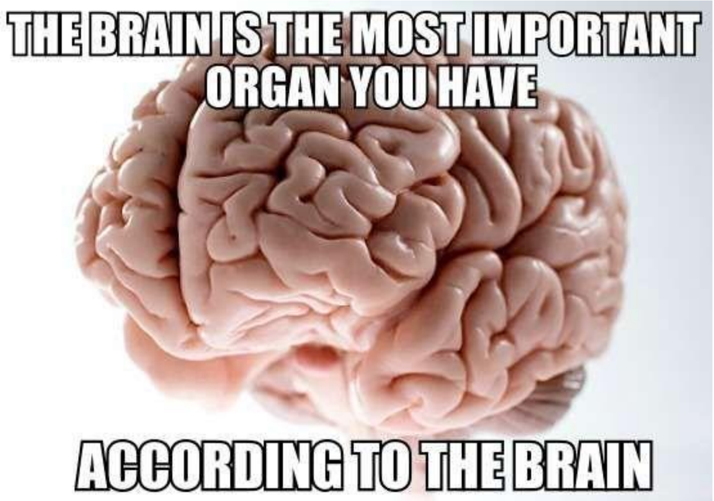
Agenda

- Biološki neuron
- Veštački neuron
- Veštačke neuronske mreže
- Dodatno čitanje

Biološki neuron

Biološki neuron

- Ljudski mozak:



Biološki neuron

- Ljudski mozak:
  - Masivni paralelizam
  - Distribuirana reprezentacija i sposobnost računanja
  - Sposobnost učenja
  - Sposobnost generalizacije
  - Prilagodljivost…

HUMAN BRAIN -v- COMPUTER

	BRAIN	COMPUTER
Size and Weight	Volume 1500 cm^3 , Weight 3.3 lbs.	Variable Weight and Size.
Construction	Neurons and Synapses	Chips, Circuits, Artificial Neurons
Structure	Bio-Genetic Programmed, Self-Learning	Pre-Programmed, AI+ML Learning to Learn
Memory	Increases by Connecting Synapses	Increased by Adding More/Better Chips
Memory Power	Teraflops (100 Trln calculations/sec)	Megabytes, Terabytes, and Zettabytes
Memory Density	10^7 circuits/cm^3	10^{14} Bits/cm^3 , and now Qubits
Info Storage	In Electrochemical and Electric Impulses	In Numeric / Symbolic Form (Binary Bits)
Info Transmission	Chemicals Fire Action Potential in Neurons	Communicates via Electrical Coded Signals
Input Tools	Human Sensory Organs	Keyboard, Mouse, Camera, Touch, Vision
Energy use	12 watts of Power; 5-10 Joules/sec	Gigawatts of Power; 10-16 Joules/sec

Sources: Various; Diagram by Frank Feather (Note: Computers are constantly advancing)

	Frontier Supercomputer	Human Brain
Speed	1.102 exaFLOPS	1 exaFlops (approximation)
Power requirements	21 MW	10-20 W
Dimensions	680 m2	1.3 to 1.4 kg
Cabling	145 km	850,000 km of axons and dendrites
Storage	58 billion transistors	125 trillion synapses, which can store 4.7 bits of information each

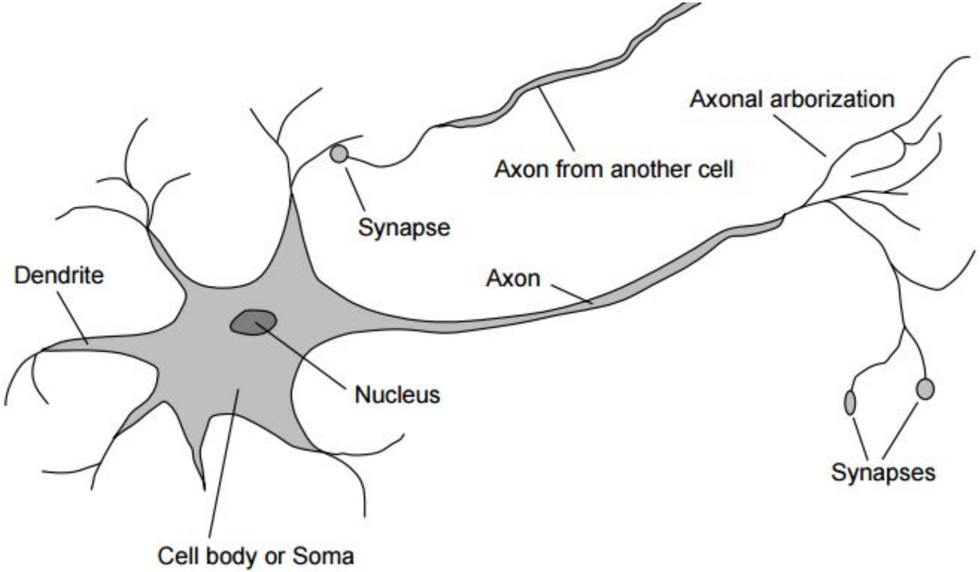
Biološki neuron

- San računarskih nauka da se napravi računar/program koji rešava izuzetno kompleksne zadatke lako i još brže nego čovek.

Biološki neuron

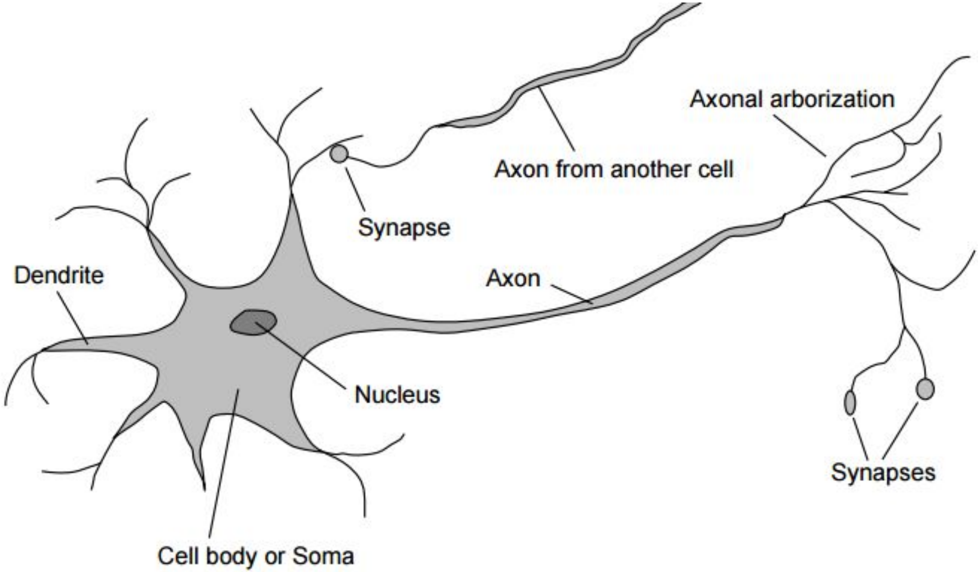
- Ljudski mozak se sastoji od oko 86,000,000,000 neurona
- Više od 20 tipova neurona

Biološki neuron



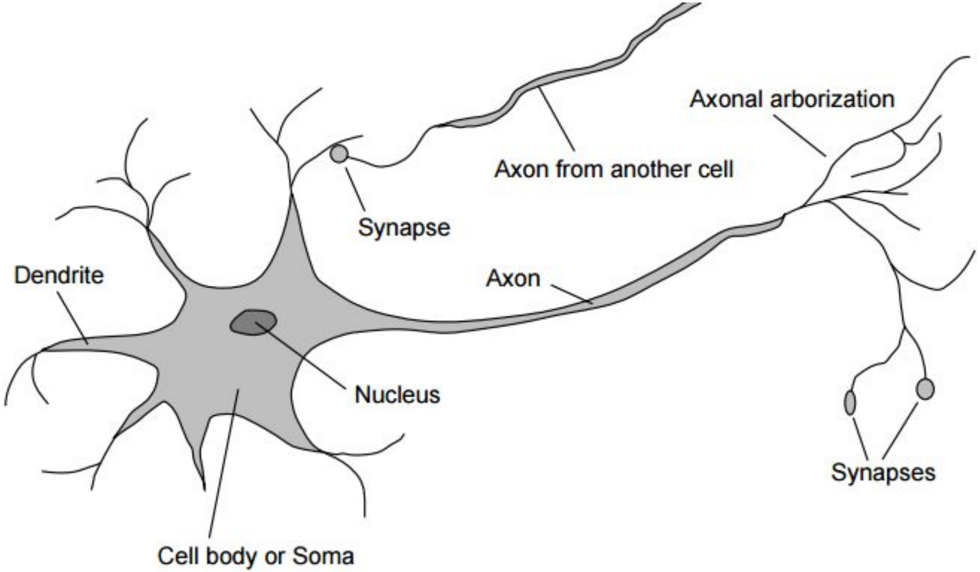
- Sinapse - veze između neurona (svaki neuron je povezan sa oko 1,000-10,000 drugih neurona):
  - Ukupno broj sinapsi ~ 10^{15}

Biološki neuron



- Impulsi do neurona dolaze preko dendrita:
  - Impulsi mogu da povećaju ili smanje verovatnoću da će neuron "ispaliti" impuls na aksonu (izlazu)
  - "Ispaljen" impuls iz aksona je ulaz u druge neurone

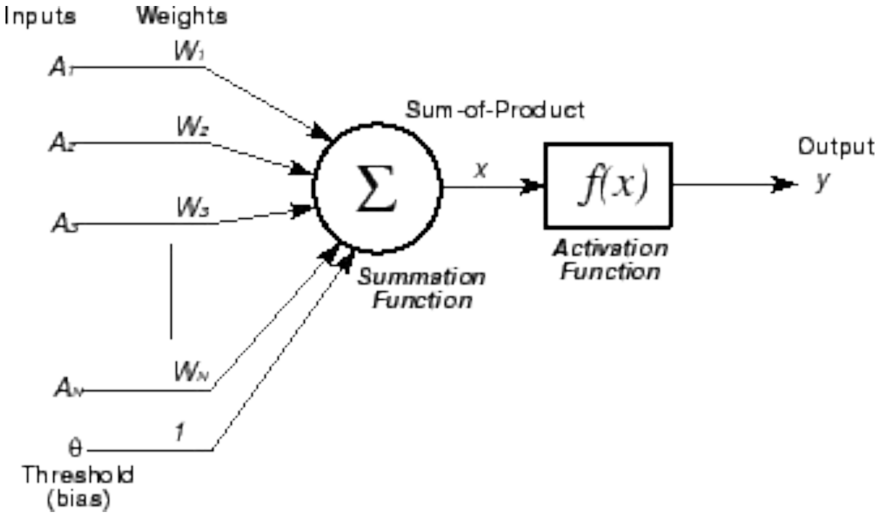
Biološki neuron



- Brzina obrade ulaznih impulsa i generisanja izlaznog impulsa je od 1ms do 10 ms:
  - Da čovek “prepozna” neku scenu/okolinu potrebno je oko 0.1s.

Veštački neuron

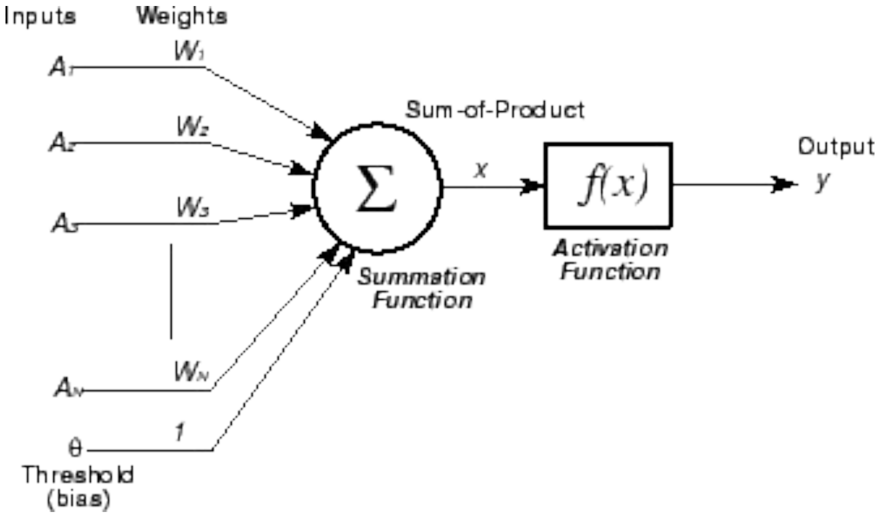
Veštački neuron



$$y = f(\sum_i^n A_i * W_i + \theta)$$

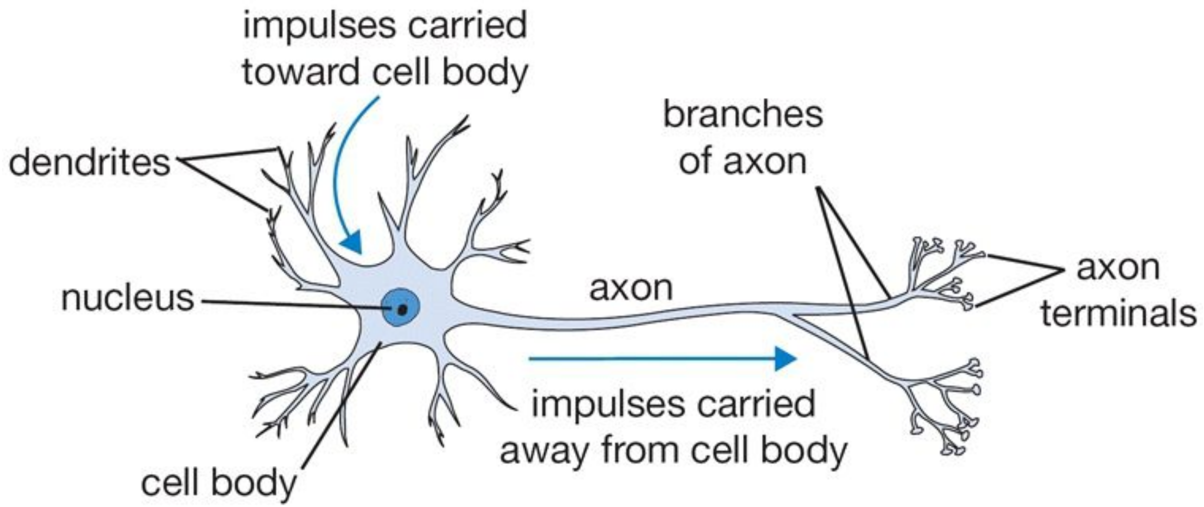
- *McCulloh-Pits* perceptron:
  - A_i - ulazi u neuron, $i \in [1, N]$
  - W_i - težine za svaki ulaz
  - θ - *bias*, konstanta nezavisna od ulaza
  - f - aktivaciona funkcija:
    - Određuje nivo aktivacije/pobuđenosti neurona za zadate ulaze
    - Postoji više aktivacionih funckija u praksi

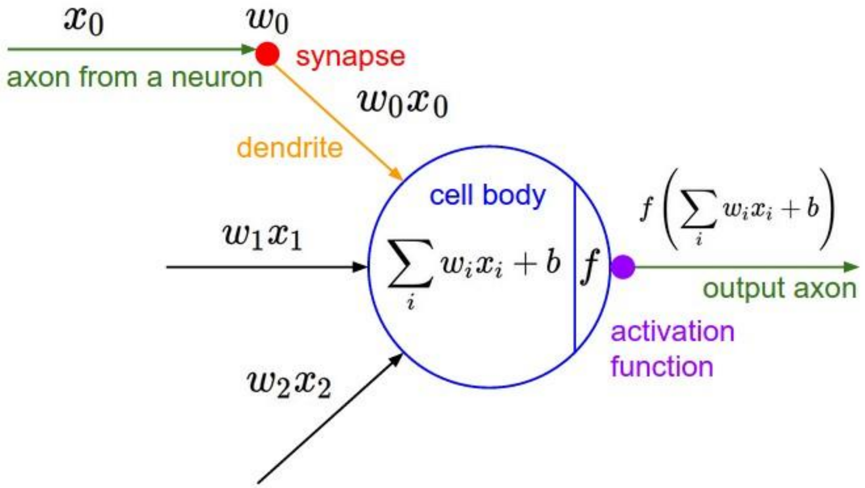
Veštački neuron



$$y = f(\sum_i^n A_i * W_i + \theta)$$

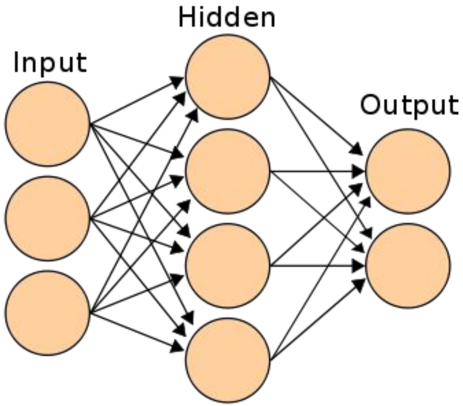
- Ogromno pojednostavljenje biološkog neurona, sa svrhom razvijanja razumevanja šta mreža ovako jednostavnih jedinica može da radi.


impulses carried
toward cell body
dendrites
branches
of axon
nucleus
axon
axon
terminals
impulses carried
away from cell body
cell body


x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation
function
$w_2 x_2$

Veštačke neuronske mreže

Veštačke neuronske mreže



- Višeslojni perceptron (eng. *Multi-layer perceptron = MLP*):
  - 1 ulazni linearni sloj (često se i ne smatra kao sloj - nesporazum u literaturi)
  - 1 izlazni (nelinearni) sloj
  - N skrivenih (nelinearnih) slojeva:
    - Formalno, ako je N > 1, u pitanju je duboka neuronska mreža

Veštačke neuronske mreže

- Obučavanje neuronske mreže:
  - Ažuriranje mreže kroz korekciju težina (*W*) tako da mreža može efikasno da izvršava željeni zadatak

Veštačke neuronske mreže

- Paradigme obučavanja:
  - Nadgledano:
    - Mrežu snabdeti ispravnim ulazima za zadate izlaze (obučavajući skup)
    - Težine se koriguju tako da mreža proizvodi sve bolje rezultate kroz obučavajući skup
  - Nenadgledano:
    - Nema potrebe zadati ispravan izlaz u obučavajućem skupu

Veštačke neuronske mreže

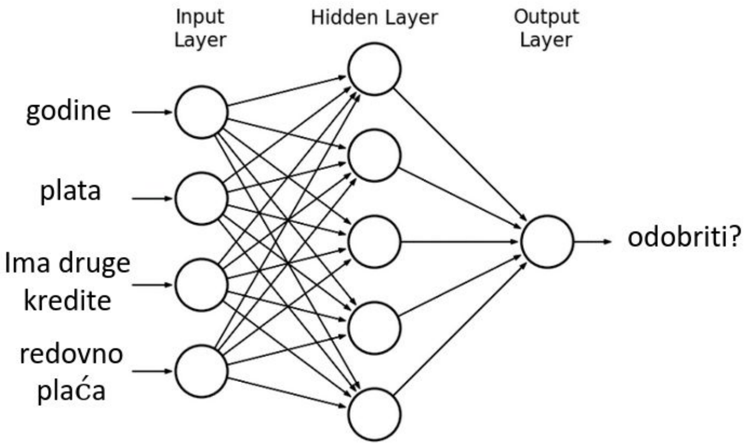
- Paradigme obučavanja:
  - Hibridno:
    - Kombinacija nadgledanog i nenadgledanog
    - Neke težine se koriguju prema ispravnom izlazu, dok se druge automatski koriguju.

Veštačke neuronske mreže

- Primer:
  - Istrenirati neuronsku mrežu koja na osnovu podataka o klijentu banke može predvideti da li tom klijentu treba odobriti stambeni kredit ili ne

Veštačke neuronske mreže

Ime i prezime	Godine	Plata	Ima druge kredite	Prethodni krediti redovno plaćani	Odobren?
Ana Anić	25	100.000	false	false	false
Pera Perić	35	150.000	false	true	true
Mika Mikić	60	45.000	true	true	false
Sara Sarić	45	35.000	true	true	false



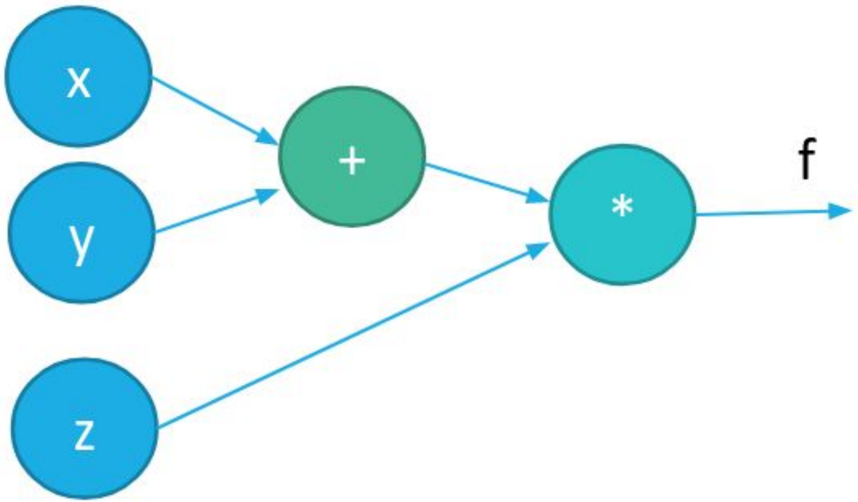
Veštačke neuronske mreže

- Ako uzmemo funkciju:
  - $f(x, y, z) = (x + y) * z$
- Kojim mehanizmom matematičke analize možemo izračunati tačno koliko svaka od nezavisnih promenljivih *x*, *y* i *z* utiče na vrednost funkcije *f*?
  - Parcijalni izvodi

$$\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \quad \frac{\partial f}{\partial z}$$

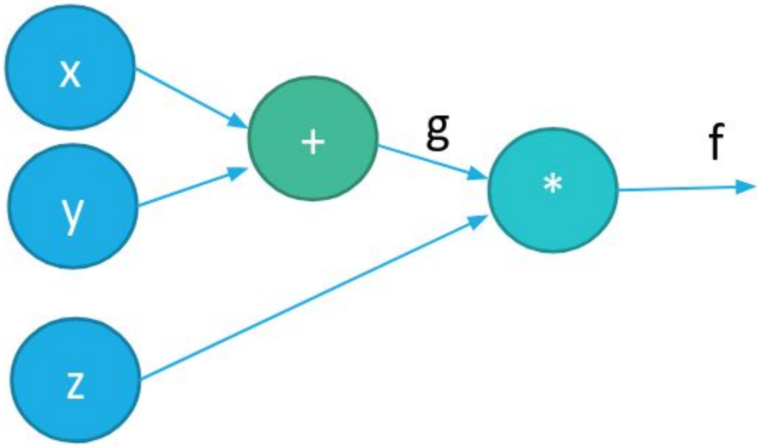
Veštačke neuronske mreže

- $f(x, y, z) = (x + y) * z$

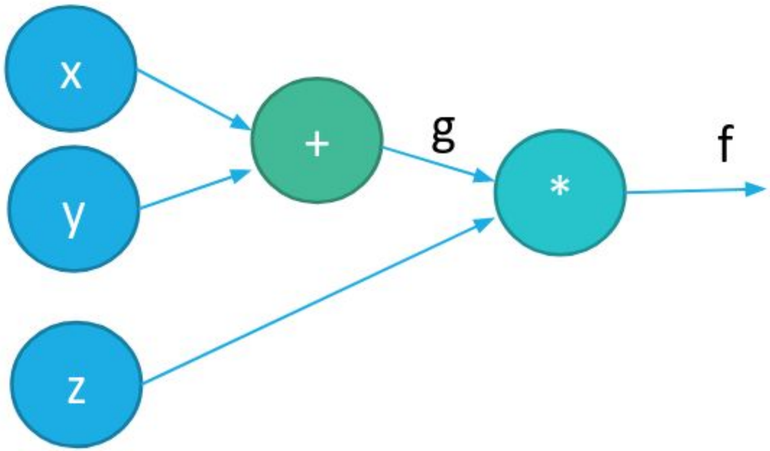


Veštačke neuronske mreže

- $f(x, y, z) = (x + y) * z$
- $g = x + y$
- $f = g * z$



Veštačke neuronske mreže



$$g = x + y$$

$$f = g * z$$

$$\frac{\partial f}{\partial z} = g = x + y$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} * \frac{\partial g}{\partial x} = z * 1 = z$$

$$\frac{\partial f}{\partial y} = \underbrace{\frac{\partial f}{\partial g} * \frac{\partial g}{\partial y}}_{\text{Pravilo ulančavanja}} = z * 1 = z$$

Pravilo ulančavanja

Pošto se parcijalni izvodi kod propagacije unazad računaju u okolini svakog pojedinačnog čvora, ovaj proces se zbog toga lokalni proces.

Veštačke neuronske mreže

$$f(x,y,z) = (x+y) * z$$

$$\frac{\partial f}{\partial x} = z$$

$$\frac{\partial f}{\partial y} = z$$

$$\frac{\partial f}{\partial z} = x + y$$

- Šta ako imamo sledeći zadatak:
  - *x = 1, y = 2, t = 12*
  - *f (x, y, z) = 12*
  - *z = ?*
- Analitičko rešavanje:
  - *(x + y) * z = 12*
  - *(1 + 2) * z = 12*
  - *z = 12/3*
  - *z = 4*

Veštačke neuronske mreže

- Međutim, šta ako je *f* funkcija sa milion nezavisnih promenljivih, a ne samo 3 i da imamo više nepoznatih?
- Koristimo numeričko rešavanje umesto analitičkog.

Veštačke neuronske mreže

- Za numerički pristup prvo moramo definisati funkciju greške, koja će nam davati informaciju koliko smo daleko ili blizu od tačnog rešenja, odnosno koliko grešimo u svakoj iteraciji, dok se polako približavamo rešenju (konvergiramo ka njemu):
  - Npr.: kvadratna funkcija greške

$$E = \frac{1}{2}(t - f)^2 \rightarrow min \quad \xrightarrow{\text{Cilj}} \quad \frac{\partial E}{\partial z} = 0$$

Veštačke neuronske mreže

$$E = \frac{1}{2}(t - f)^2 \qquad \frac{\partial E}{\partial z} = 0$$

$$\frac{\partial E}{\partial z} = \frac{\partial E}{\partial f} * \frac{\partial f}{\partial z} = \frac{1}{2} * 2(t - f) * (-1) * (x + y)$$

$$= (f - t)(x + y) = (f - 12) * 3$$

Veštačke neuronske mreže

$$\frac{\partial E}{\partial z} = (f - 12) * 3$$

- Šta sad da radimo sa ovim izvodom?
- Izvod nam pokazuje tačno koliko neka konkretna promenljiva utiče na grešku.
- Intenzitet tog izvoda možemo iskoristiti da promenimo vrednost te konkretne promenljive, da bi u sledećoj iteraciji stvarala manju grešku.

Veštačke neuronske mreže

- Prethodno smo definisali funkciju greške, kojom vršimo procenu greške (*forward*) i uticaja određene promeljive na grešku putem njenog izvoda po toj promenljivoj (*backward*)
- Da bismo modifikovali vrednost promenljive i iskoristili izračunati izvod za smanjenje greške u sledećim iteracijama, moramo definisati optimizacioni algoritam, čijom se formulom vrši modifikacija promenljive.

Veštačke neuronske mreže

- *Gradient Descent* algoritam (GD):

$$z_{novo} = z_{staro} - \alpha * \frac{\partial E}{\partial z}$$

Veštačke neuronske mreže

$$f(x, y, z) = (x + y) * z$$

$$\frac{\partial E}{\partial z} = (f - 12) * 3$$

$$z_{novo} = z_{staro} - \alpha * \frac{\partial E}{\partial z}$$

- Nasumično izaberemo vrednost nepoznate *z* (početno pogađanje) i krećemo u numerički algoritam:

Veštačke neuronske mreže

- Iteracija 1:

$$z = 1$$

$$f = (1 + 2) * 1 = 3$$

$$\frac{\partial E}{\partial z} = (3 - 12) * 3 = -27$$

$$z = 1 - 0.1 * (-27) = 1 + 2.7 = 3.7$$

Veštačke neuronske mreže

- Iteracija 2:

$$z = 3.7$$

$$f = (1 + 2) * 3.7 = 11.1$$

$$\frac{\partial E}{\partial z} = (11.1 - 12) * 3 = -2.7$$

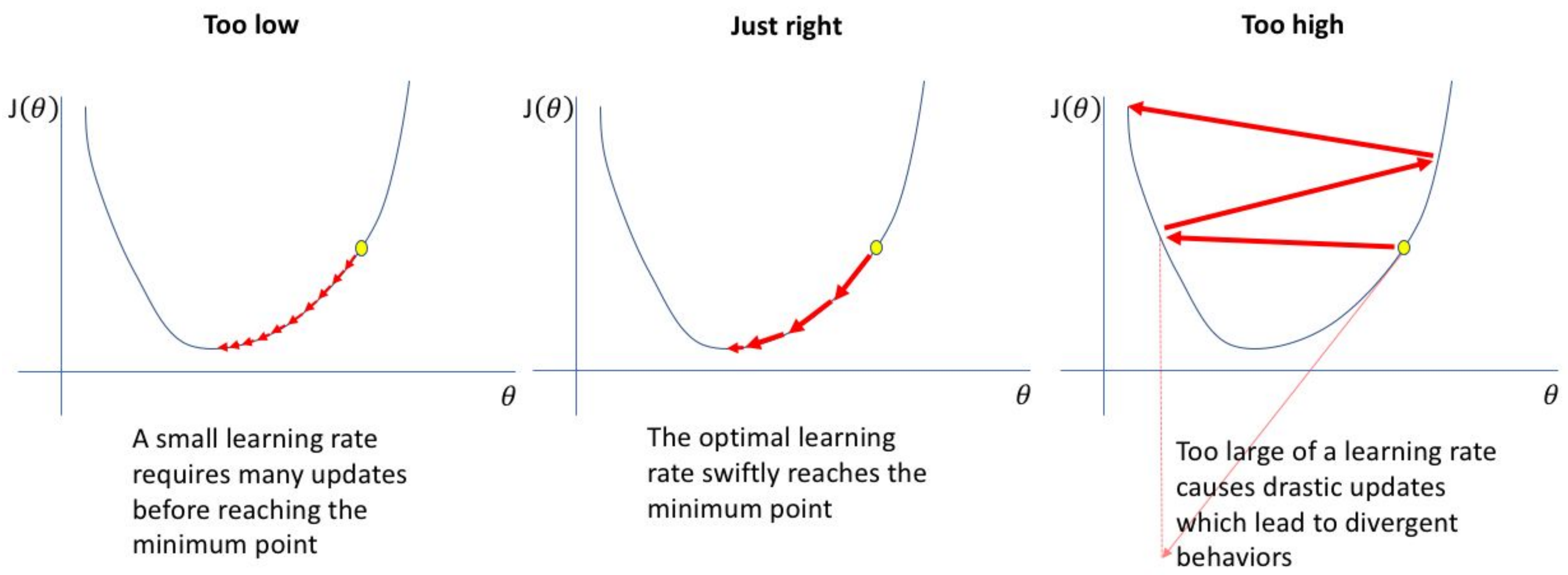
$$z = 3.7 - 0.1 * (-2.7) = 3.7 + 0.27$$

$$z = 3.97$$

Veštačke neuronske mreže

- Parametar α se obično postavlja na vrednost 10^{-1} ili 10^{-2}, a nakon toga se iterativno smanjuje
- Na taj način se grublje dolazi do dela prostora u kome se nalazi rešenje, a onda se vrednost smanjuje kako bi došli do finijeg rešenja
- U praksi se obično koristi 10^{-2}, ali opet sve zavisi od prirode problema koji se rešava.

Veštačke neuronske mreže



Veštačke neuronske mreže

- Nadgledano obučavanje neuronske mreže:
  - Minimizovati grešku tokom obučavanja tako da izlazi mreže što više odgovaraju željenim ulazima:
    - Hoćemo da je kvadratna greška što manja

$$E(W) = \frac{1}{2}\sum_{i=1}^{N}(y_i - o_i)^2$$

Veštačke neuronske mreže

- Nadgledano obučavanje neuronske mreže:
  - Problem: unapred ne znamo “ispravne” izlaze neurona u skrivenom sloju
    - Što zapravo i nije strašno - možemo koristiti klasični GD

Veštačke neuronske mreže

$$W = W - \alpha * \nabla W$$

- Za svaku težinu računa se parcijalni izvod funkcije greške u odnosu na tu težinu
- Težina se koriguje tako što se ide kontra od smera izvoda sa nekim korakom *α* (tzv. brzina obučavanja)

$$W_{i,j} = W_{i,j} - \alpha * \frac{\partial E}{\partial W_{i,j}}$$

Veštačke neuronske mreže

- Nadgledano obučavanje neuronske mreže:
  - Navedeni pristup je u redu ako je u pitanju jednoslojni perceptron
  - Šta ćemo sa višeslojnim?
    - *Backpropagation*

Veštačke neuronske mreže

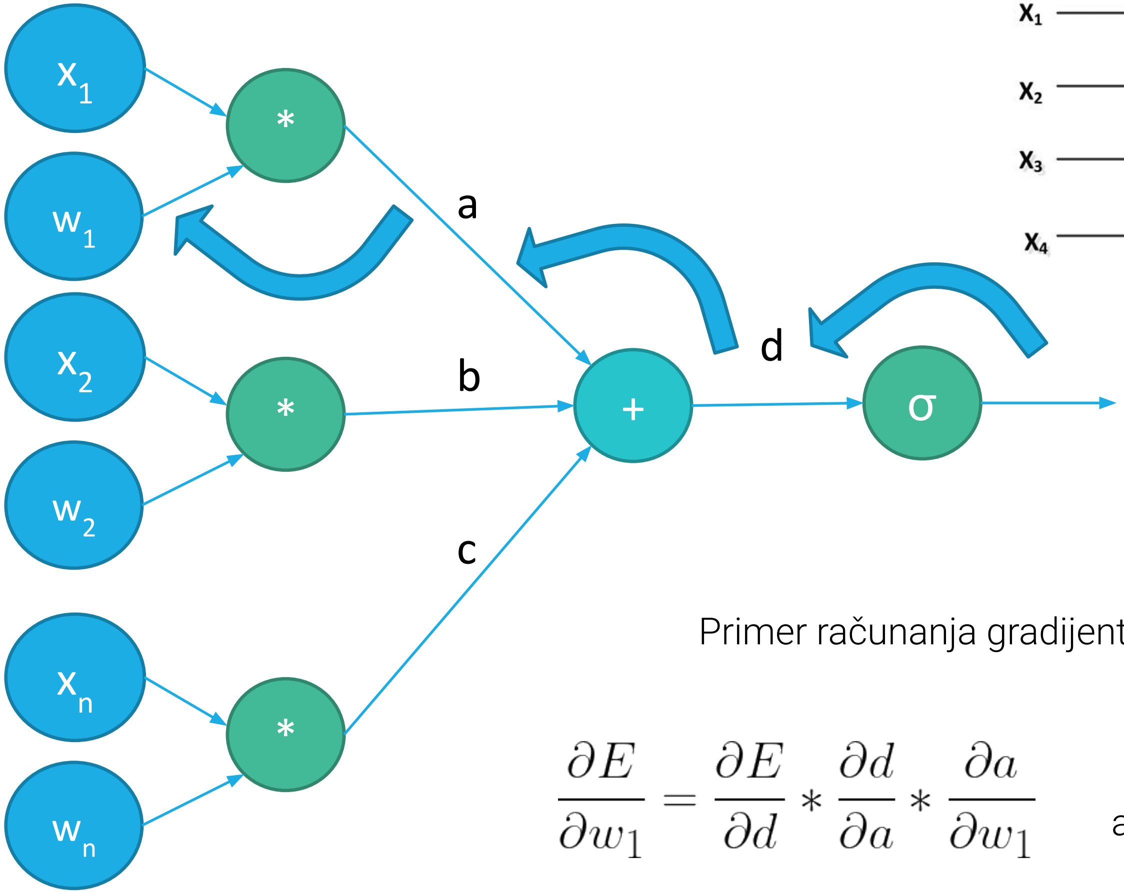
- *Backpropagation*:
  - Za svaki parametar u obučavajućem skupu:
    - Propagirati ulazni signal unapred kroz neuronsku mrežu (eng. *forward pass*)
    - Propagirati greške unazad kroz neuronsku mrežu:
      - Za svaku težinu u mreži izračunati $\delta E / \delta W_{i,j} = \delta_j * o_i$
      - $\delta_j = (o_j - y_j) * o_j * (1 - o_j)$ za neurone u izlaznom sloju
      - $\delta_j = (\Sigma_{l \in L} \delta_l * W_{j,l}) * o_j * (1 - o_j)$ za neurone u skrivenim slojevima
    - Korigovati težine po formuli $W_{i,j} = W_{i,j} - \Delta W_{i,j} = W_{i,j} - \alpha * \delta E / \delta W_{i,j}$
  - Ovo ponavljati N epoha

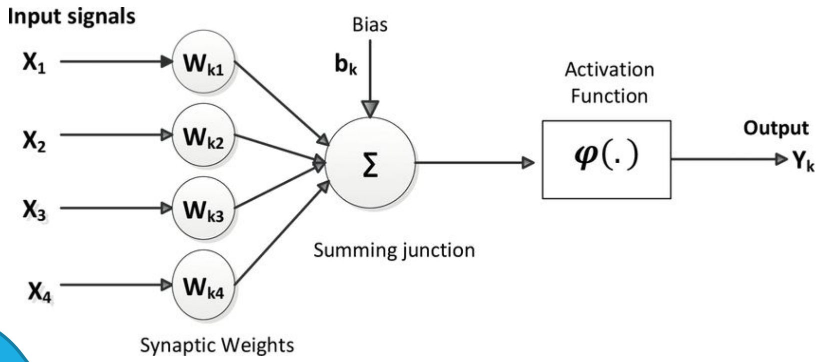
Veštačke neuronske mreže

- *Gradient Descent (GD):*
  - Jedna epoha obučavanja - uvek se prolazi kroz ceo obučavajući skup
  - Ako je obučavajući skup ogroman (što je obično slučaj), ovo ne dolazi u obzir
  - Deterministički postupak, svaki put će mreža biti identično obučena

Veštačke neuronske mreže

- *Stochastic Gradient Descent (SGD):*
  - U svakoj epohi obučavanja - uzima samo N nasumičnih primeraka iz obučavajućeg skupa
  - Brže konvergira od GD
  - Stohastičan postupak, zbog nasumičnosti, mreža će svaki put biti drugačije obučena
  - Uglavnom bude jako dobra aproksimacija rezultata dobijenih sa GD
  - U praktičnoj primeni, SGD se uvek koristi umesto GD





Primer računanja gradijenta za težinu W_1 pravilom ulančavanja:

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial d} * \frac{\partial d}{\partial a} * \frac{\partial a}{\partial w_1}$$

Loss

a onda ažuriranje

$$w_1 = w_1 - \alpha * \frac{\partial E}{\partial w_1}$$

Optimize

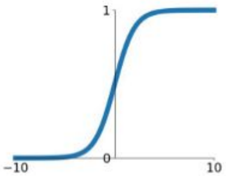
Veštačke neuronske mreže

- Aktivaciona funkcija:
  - Osnovna namena im je unošenje nelinearnosti u sistem

Veštačke neuronske mreže

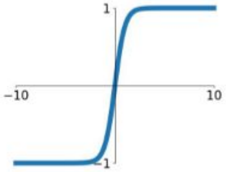
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



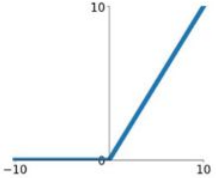
tanh

$\tanh(x)$



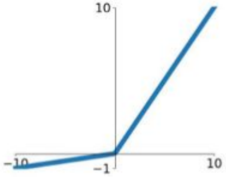
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

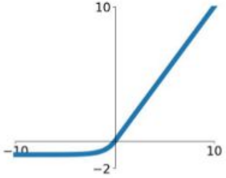


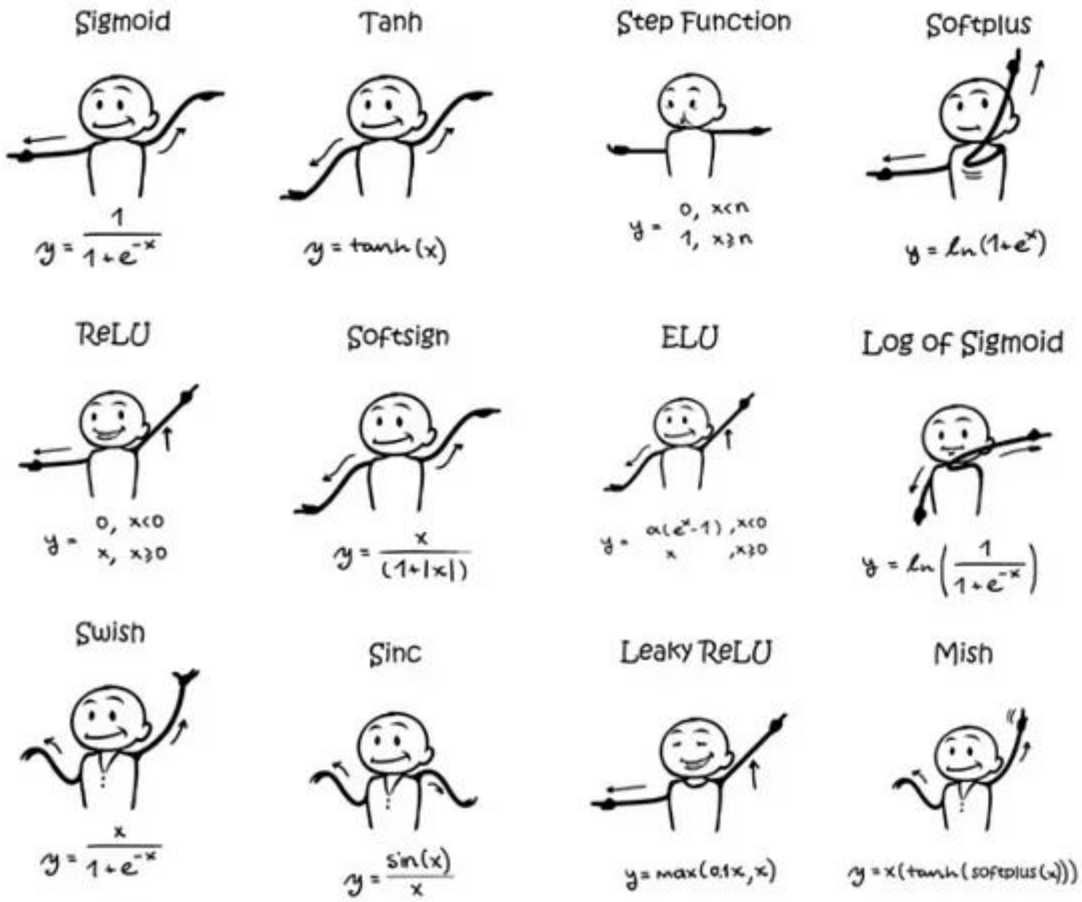
Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

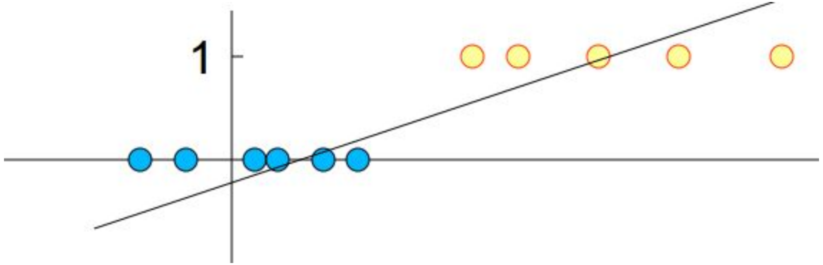




Sigmoid
y = 1 / (1 + e^-x)
Tanh
y = tanh(x)
Step Function
y = 0, x<n
1, x≥n
Softplus
y = ln(1+e^x)
ReLU
y = 0, x<0
x, x≥0
Softsign
y = x / (1+|x|)
ELU
y = α(e^x-1), x<0
x, x≥0
Log of Sigmoid
y = ln(1 / (1+e^-x))
Swish
y = x / (1+e^-x)
Sinc
y = sin(x) / x
Leaky ReLU
y = max(0.1x, x)
Mish
y = x(tanh(softplus(x)))

Veštačke neuronske mreže

- Linearni perceptron:
  - Samo *y* = *Wx*, gde je *x* ulazni vektor sa dodatim $x_0 = 1$ (*bias*)
  - Može da predstavi bilo koju linearnu funkciju u D + 1 dimenzionalnom prostoru
  - U suštini radi najobičniju linearnu regresiju
  - Nije zgodan za modelovanje binarne klasifikacije



Veštačke neuronske mreže

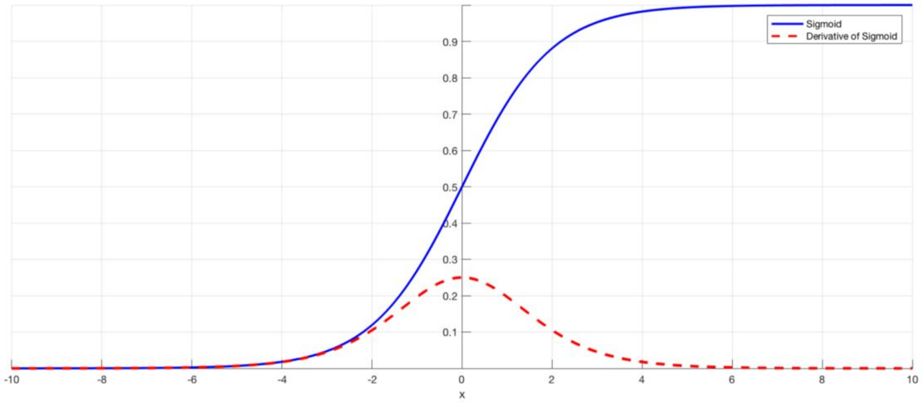
- Nelinearni perceptron:
  - Može da modeluje linearno neseparabilne probleme:
    - Što u teoriji znači da može da modeluje bilo šta (čak i neke nekontinualne funkcije)
  - Sigmoidalne funkcije su zgodne jer:
    - Imaju linearni deo (otprilike u opsegu $x \in [-1, 1]$)
    - Ali imaju i nelinearni deo kojim se može modelovati neka klasifikacija (0, 1)

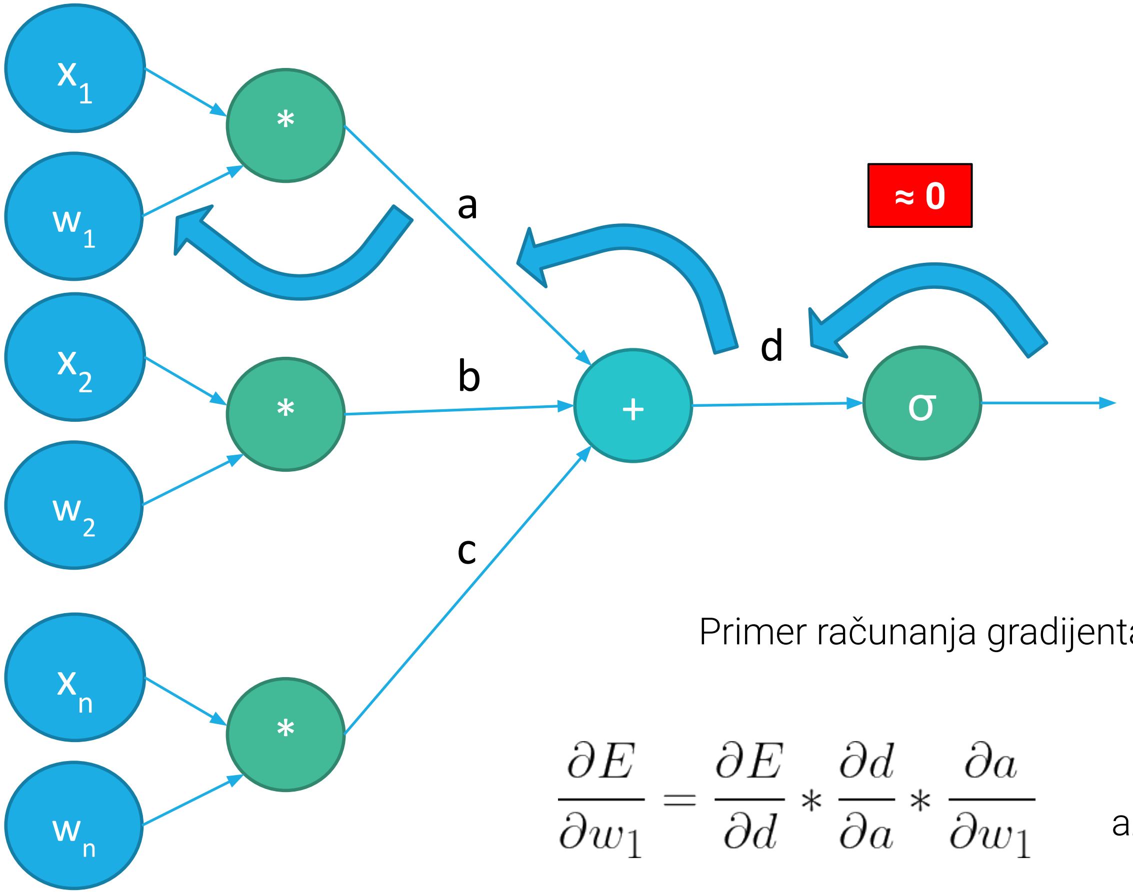
Veštačke neuronske mreže

- Aktivaciona funkcija - *Vanishing gradient*:
  - Šta ako izlaz sabirača jednog neurona bude jako velika vrednost po apsolutnoj vrednosti?

Veštačke neuronske mreže

- Aktivaciona funkcija - *Vanishing gradient*:
  - Dolazimo do situacije da parcijalni izvod oko sigmoidnog čvora (u grafu izračunavanja) u tom slučaju ima vrednost koja teži nuli.





Mreža se obučava, ali esktremno sporo.

Takođe, kod višeslojnih mreža se najsporije obučavaju početni slojevi, jer do njih gradijent nestane još više nego za slojeve blizu izlaza (duža jednačina ulančavanja).

Primer računanja gradijenta za težinu W_1 pravilom ulančavanja:

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial d} * \frac{\partial d}{\partial a} * \frac{\partial a}{\partial w_1}$$

≈ 0

a onda ažuriranje

$$w_1 = w_1 - \alpha * \frac{\partial E}{\partial w_1}$$

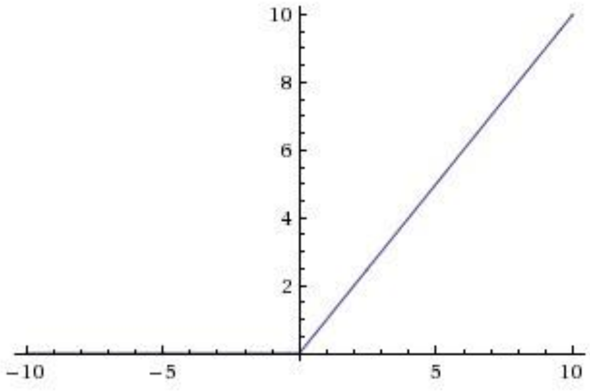
≈ 0

Veštačke neuronske mreže

- Aktivaciona funkcija - *Vanishing gradient*:
  - Potencijalna rešenja:
    - Normalizacija vrednosti ulaznih podataka u mrežu
    - Korišćenje neke druge aktivacione funkcije (ne sigmoidne)

Veštačke neuronske mreže

- *ReLU* aktivaciona funkcija:
  - ***Re****ctifying* ***L****inear* ***U****nit*
  - *f(x) = max(0, x)*
  - Trenutno najpopularnija aktivaciona funkcija za neurone u skrivenom sloju



Veštačke neuronske mreže

- *ReLU* aktivaciona funkcija:
  - Zašto?
    - Lako se računa
    - Nema *vanishing gradient* problem kao *sigmoid* i *tanh*
    - Brže obučavanje
  - Varijacije:
    - *RReLU*
    - *Leaky ReLU*

Veštačke neuronske mreže

- *Softmax* aktivaciona funkcija:
  - Koristi se za izlazni sloj kada se vrši multiklasna klasifikacija
  - Na izlaznom sloju, za razliku od sigmoidalnog sloja koja modeluje N od M klasifikaciju (za određeni ulaz moguće je da bude aktivirano više izlaza), *Softmax* sloj modeluje 1 od M klasifikaciju (za određeni ulaz moguć je samo jedan izlaz)
  - Suma izlaza iz mreže je 1

Veštačke neuronske mreže

- *Softmax* aktivaciona funkcija:

$$P\left(y^{(i)} = 1 | x^{(i)}, \theta\right) = \frac{e^{\theta x^{(i)}}}{\sum_{j=1}^{M} e^{\theta x^{(j)}}}$$

Veštačke neuronske mreže

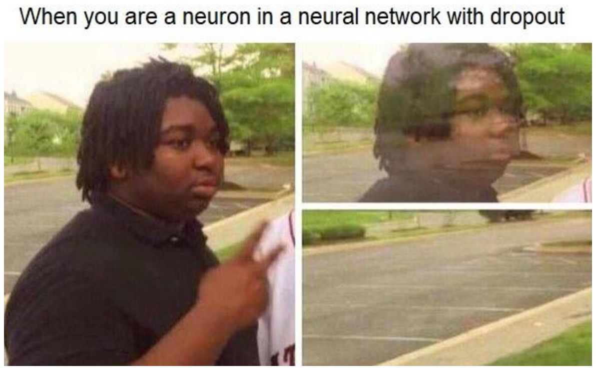
- Problemi obučavanja neuronske mreže (1/2):
  - *Overfitting*:
    - Mreža previše dobro obučena na obučavajućem skupu
    - Nema sposobnost generalizacije
    - Radi ispravno samo na podacima koje je već videla
  - *Underfitting*:
    - Mreža uopšte nike dobro obučena
    - Jednostavno težine još nisu konevrgirale
  - Kada prekinuti obučavanje?
    - *Early stopping, checkpoints*...

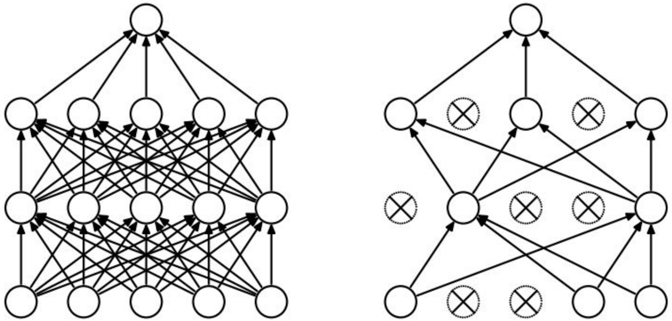
Veštačke neuronske mreže

- Dropout:
  - Jedan od najvećih izuma na polju neuronskih mreža u poslednje vreme (2014)
  - Protiv *overfitting*-a - dramatično bolje (i jednostavnije) od nekih drugih metoda

Veštačke neuronske mreže

- Dropout:
  - Tokom obučavanja nasumično ugasiti/izbaciti neurone
  - U svakoj epohi obučavanja nasumično se biraju neuroni koji će biti ugašeni/izbačeni samo u toj epohi





Veštačke neuronske mreže

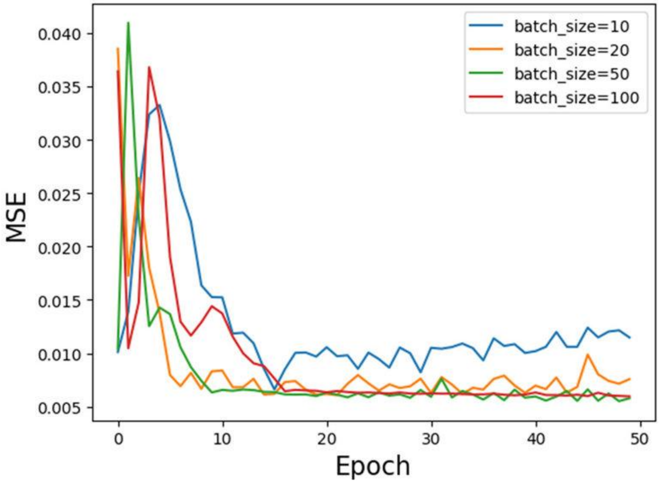
- Dropout:
  - Sprečava *overfitting* tako što sprečava neurone da se previše adaptiraju obučavajućem skupu (jer u svakoj epohi postoji šansa da će biti izbačeni)
  - Može se posmatrati i kao da u svakoj epohi obučavamo drugu mrežu, i samim tim kao da je konačno obučena mreža skup više jednostavnijih mreža
  - Uobičajene vrednosti su između 20% i 50%

Veštačke neuronske mreže

- Problemi obučavanja neuronske mreže (2/2):
  - Lokalni optimumi:
    - Teže konvergiraju ka lokalnom minimum

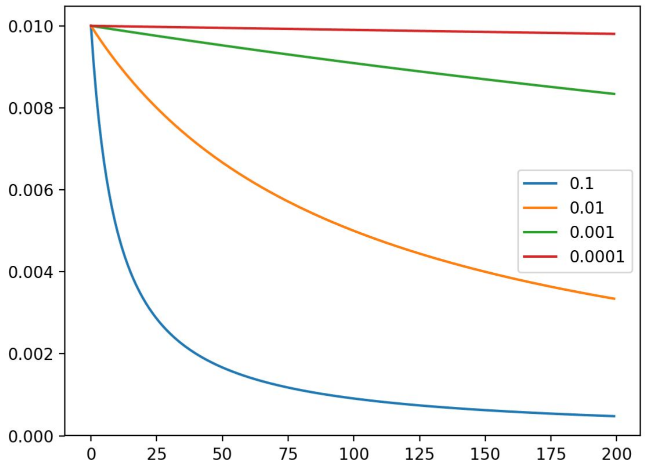
Veštačke neuronske mreže

- Problemi obučavanja neuronske mreže (2/2):
  - *Batch size*:
    - Nakon koliko primeraka ažurirati težine (nakon svakog, nakon svih ili negde između)



Veštačke neuronske mreže

- Problemi obučavanja neuronske mreže (2/2):
  - Brzina obučavanja:
    - Konvergencija izuzetno osetljiva na brzinu obučavavanja (izuzetno sporo ili preskakanje minimuma)



Veštačke neuronske mreže

- Još problema?
  - Koliko neurona u skrivenim slojevima?
  - Koliko skrivenih slojeva?
  - Koja funkcija greške?
  - Koji su parametri obučavanja?
  - Koja aktivaciona funkcija?
  - Kako povezati neurone?

Veštačke neuronske mreže

- Primer:
  - *Backpropagation* uz graf izračunavanja

Dodatno čitanje

Dodatno čitanje

- Istorijat i modelovanje
- Priprema podataka i funkcije greške
- *Backpropagation*
- Jednostavno izvođenje *Backpropagation* algoritma

Hvala na pažnji!

Pitanja?