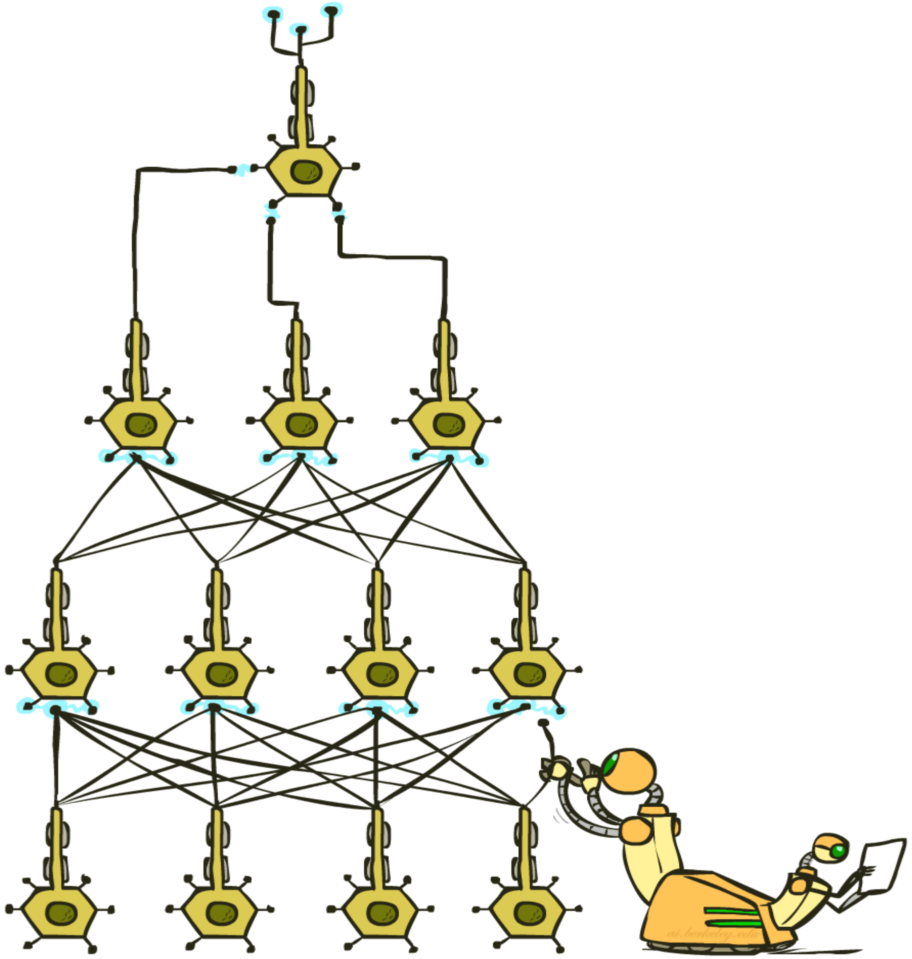
Osnovi Računarske Inteligencije

Uvod u Neuronske Mreže



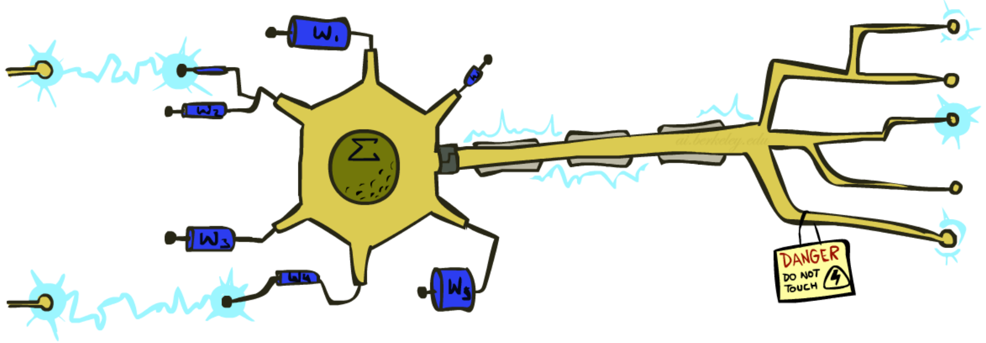
Predavač: Aleksandar Kovačević

Slajdovi preuzeti sa kursa CS188, University of California, Berkeley

http://ai.berkeley.edu/

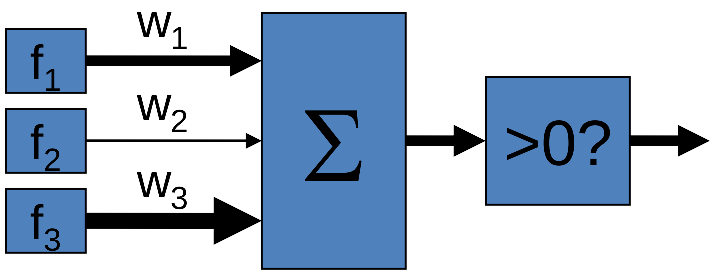
Prošli put: Perceptron

- Ulazi su vrednosti osobina
- Svakoj osobini dodeljena je težina
- Suma je aktivacija

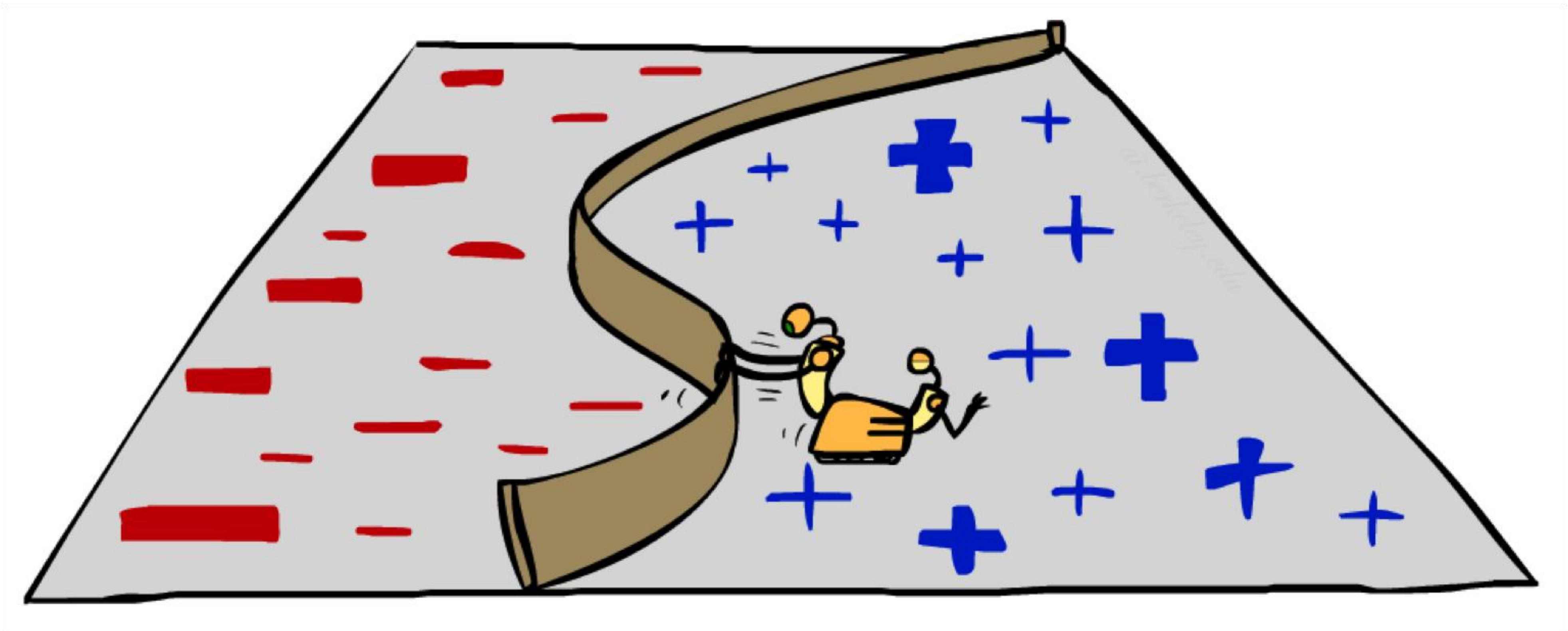


$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- Ako je aktivacija:
  - Pozitivna, izlaz je +1
  - Negativna, izlaz je -1

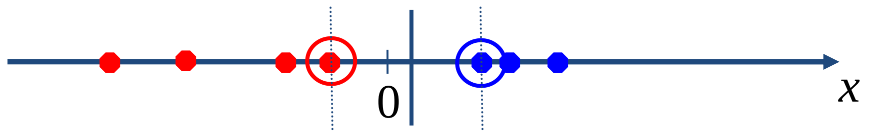


Ne-linearna Granica Odluke

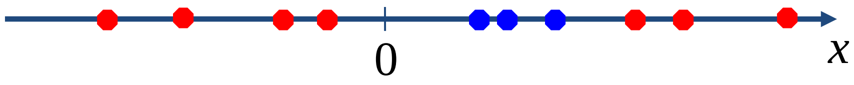


Ne-linearni Klasifikatori

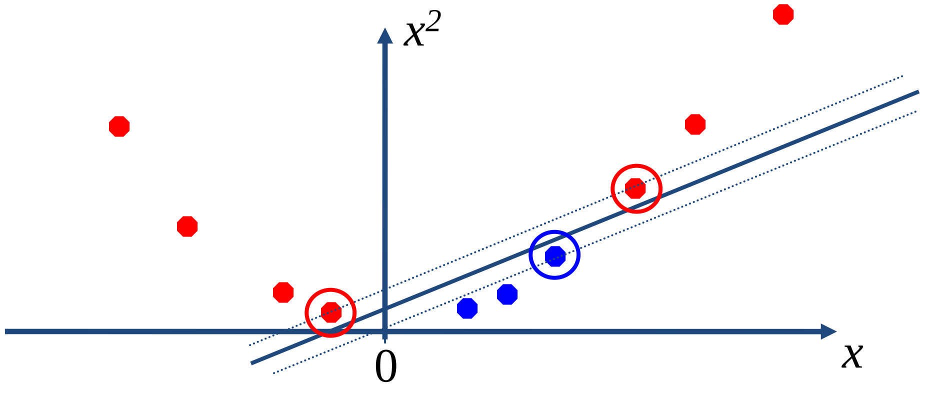
- Ako su podaci linearno razvojivi, percetron i slični modeli (SVM) rade dobro:



- Šta ćemo da uradimo ako dobijemo stvano težak klasifikacioni problem?



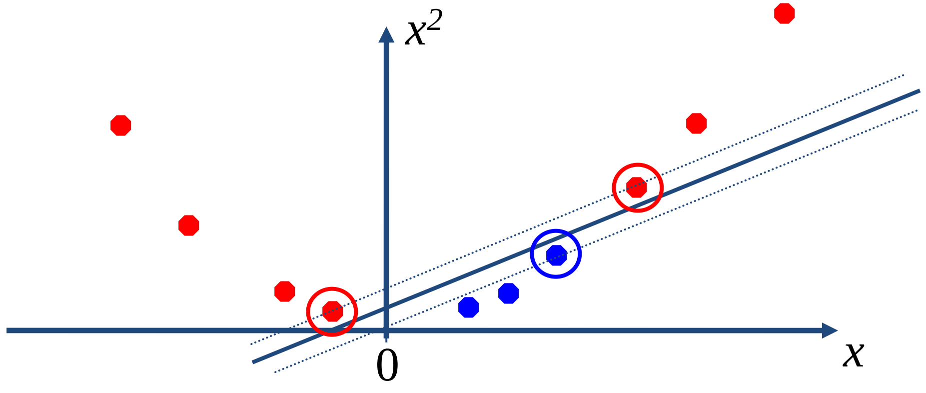
- Da li možemo da mapiramo podatke na prostor u kome su linearno razdvojivi?:



This and next slide adapted from Ray Mooney, UT

Ne-linearni Klasifikatori

- Da li možemo da mapiramo podatke na prostor u kome su linearno razdvojivi?:

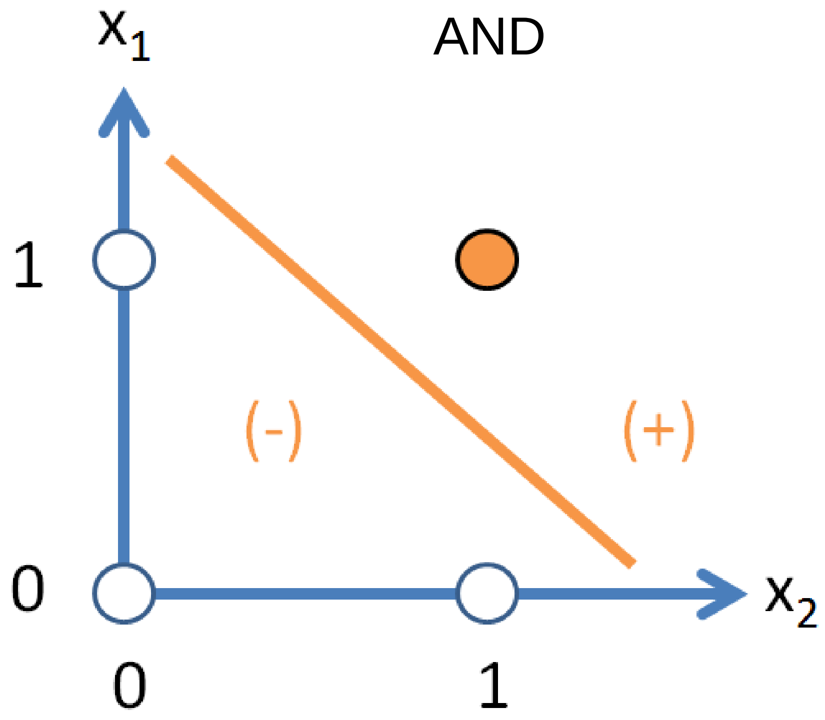


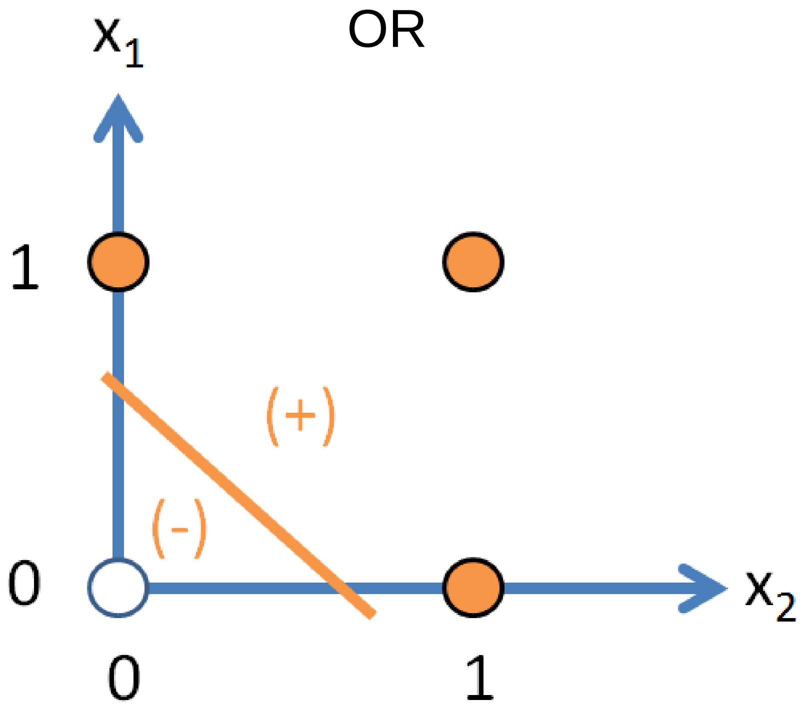
- Ovo jeste opcija i radićete je na više predmeta kasnije, ali
- nije uvek lako odrediti način mapiranja, uz to postoje i mnoge druge mane koje nećemo sad navoditi (npr. *overfitting*)

- Danas radimo Neuronske Mreže koje su idealna alternativa!

Ne-linearni Klasifikatori

- Potrebu za Neuronskim mrežama kao ne-linearnim klasifikatorima objasnićemo pomoću Perceptrona
- Konkretno realizovaćemo logičke operacije AND i OR koristeći perceptron.
- Logičko AND i OR su linearno separabilini klasifikacioni problemi.





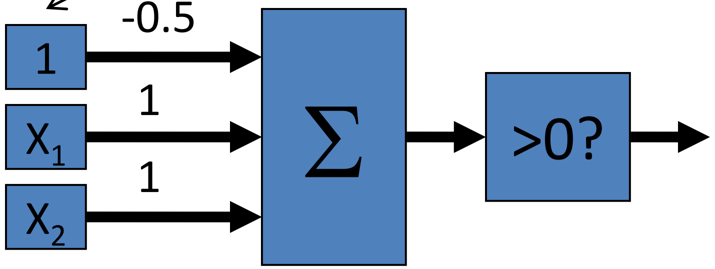
Izvor slika: http://web.cs.ucla.edu/~forns/classes/winter-2016/cs-161/week-10.html

Ne-linearni Klasifikatori

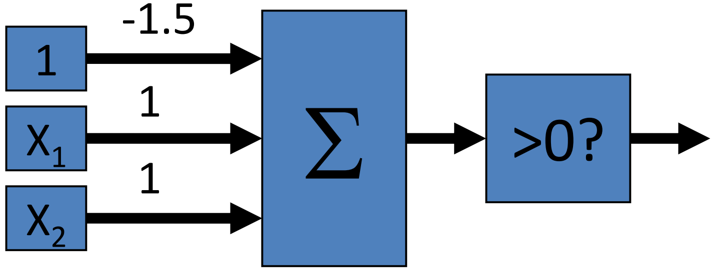
- Logičko AND i OR su linearno separabilini klasifikacioni problemi.
- Dakle svaka od ove dveoperacije može se realizovati jednim perceptronom

Neuron koji je uvek uključen (aktivacija uvek =1), ekvivalent b delu iz linearnih modela (wx+b)

OR

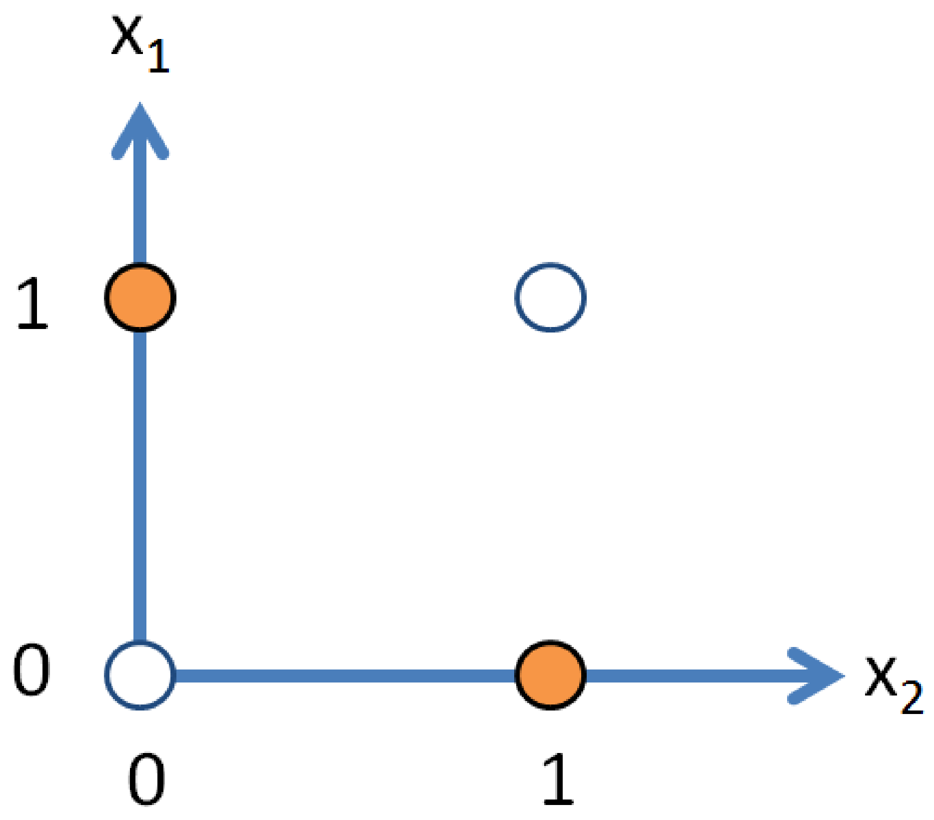


AND



Ne-linearni Klasifikatori

- Da li pomoću jednog perceptrona možemo da realizujemo logičku operaciju XOR?



Ne-linearni Klasifikatori

- Vidimo da XOR nije linearno separabilan problem, tako da nam jedan perceptron nije dovoljan.

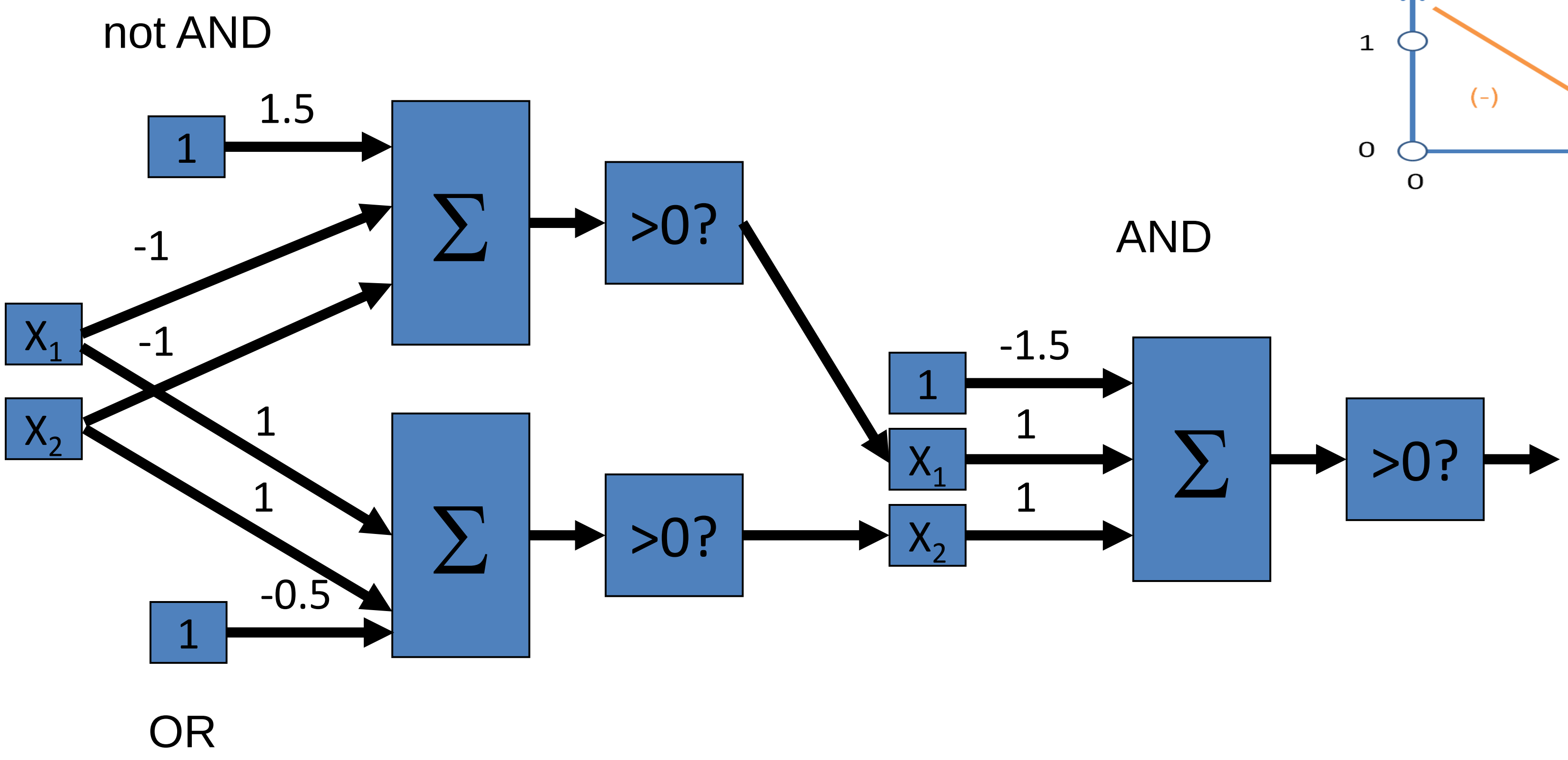
- Rešenje: Upotrebićemo više perceptrona!

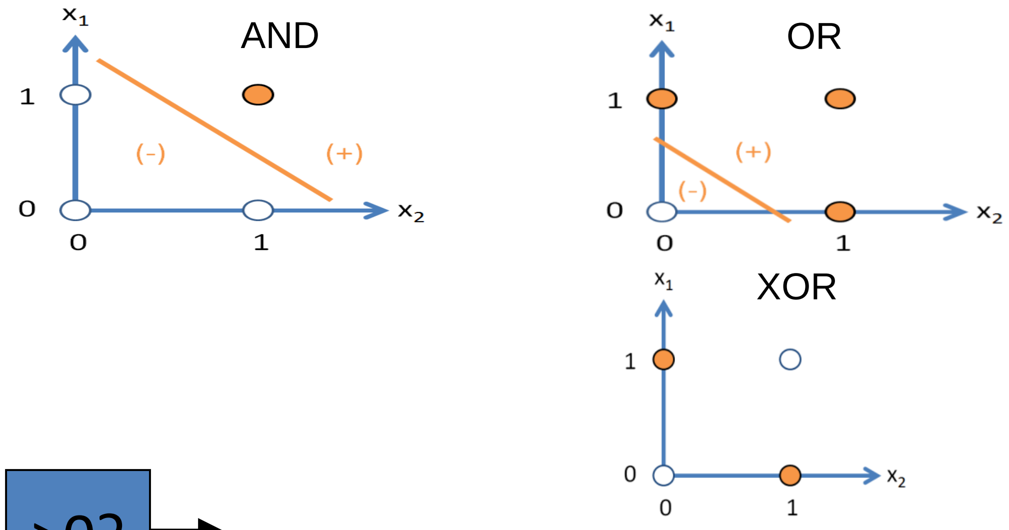
- Perceptrone ćemo spojiti u slojeve, izlaz jednog biće ulaz u drugi.

- Takav model zove se Višeslojni Perceptron (*Multilayer Perceptron*, MLP)

Ne-linearni Klasifikatori

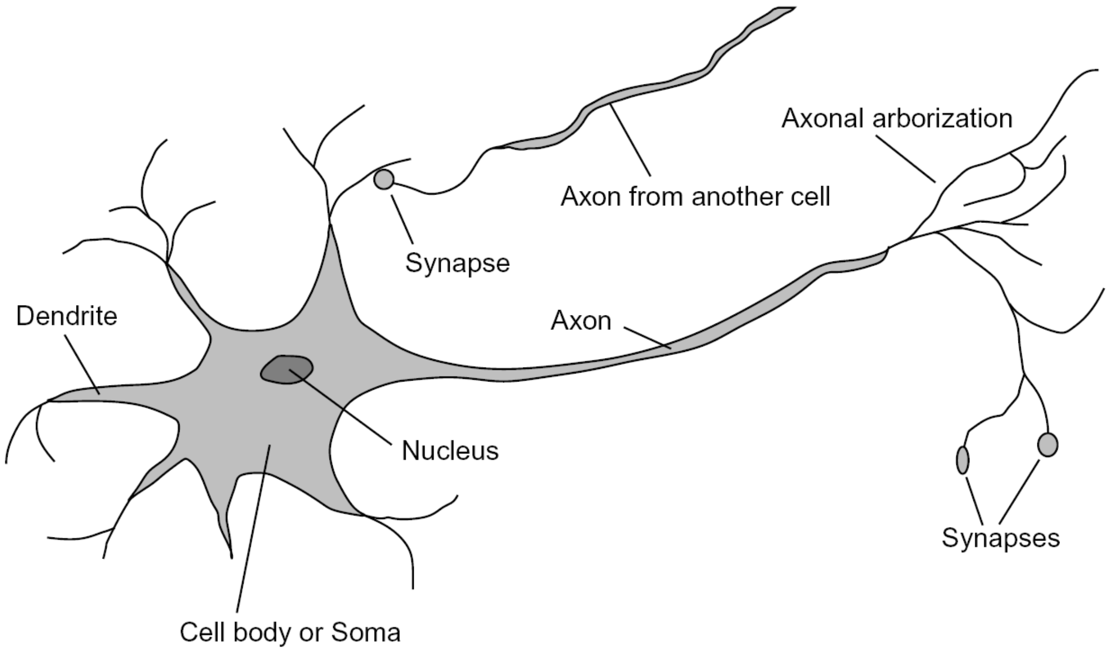
- Višeslojni perceptron za XOR problem
- XOR(x,y) = OR(x,y) AND not AND (x,y)

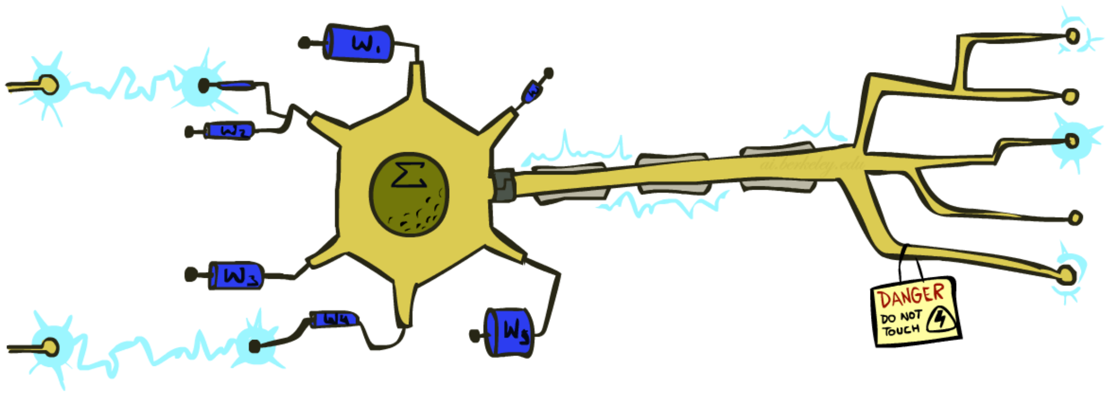




Neuronska Mreža

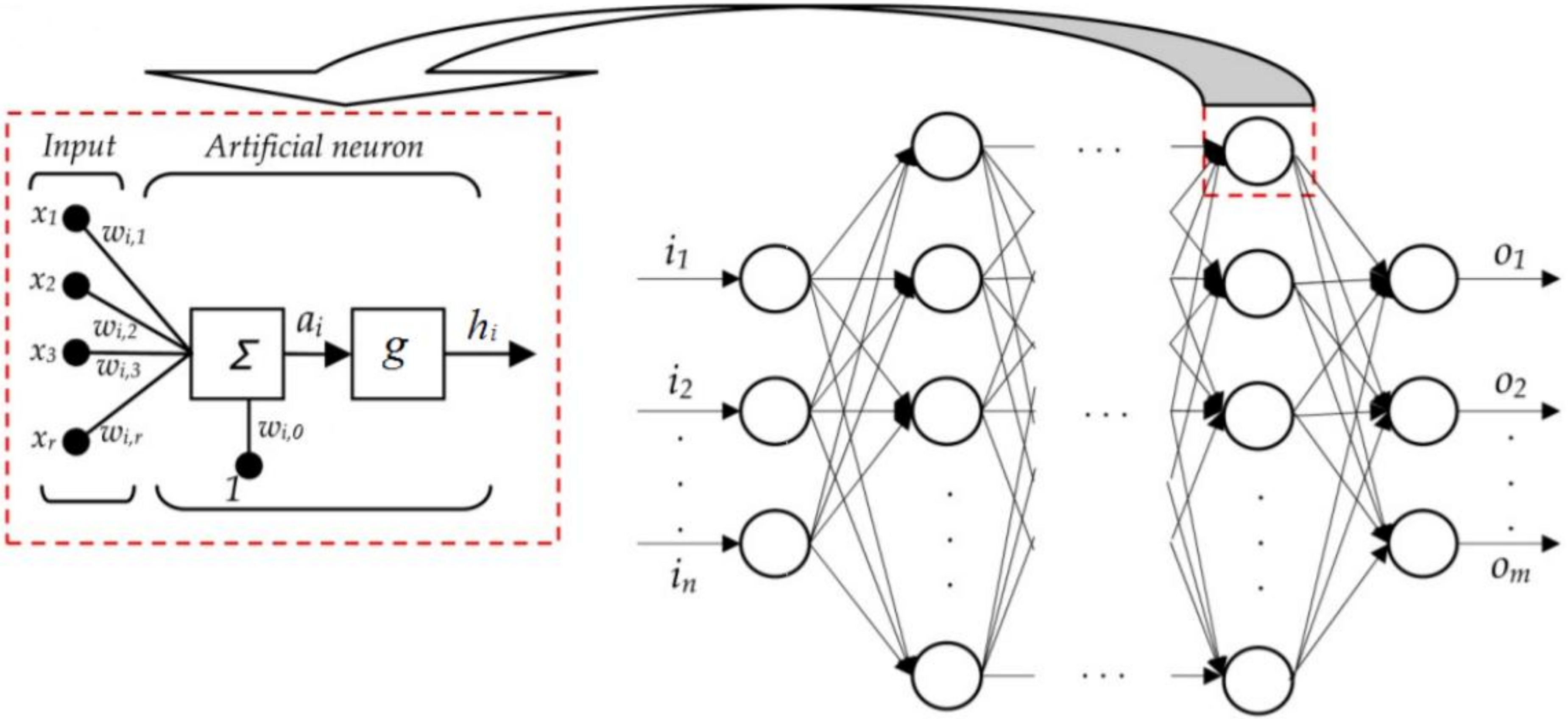
- Ako je perceptron jedan neuron, šta je onda Višeslojni Perceptron?
- Višeslojni perceptron je onda Neuronska Mreža!



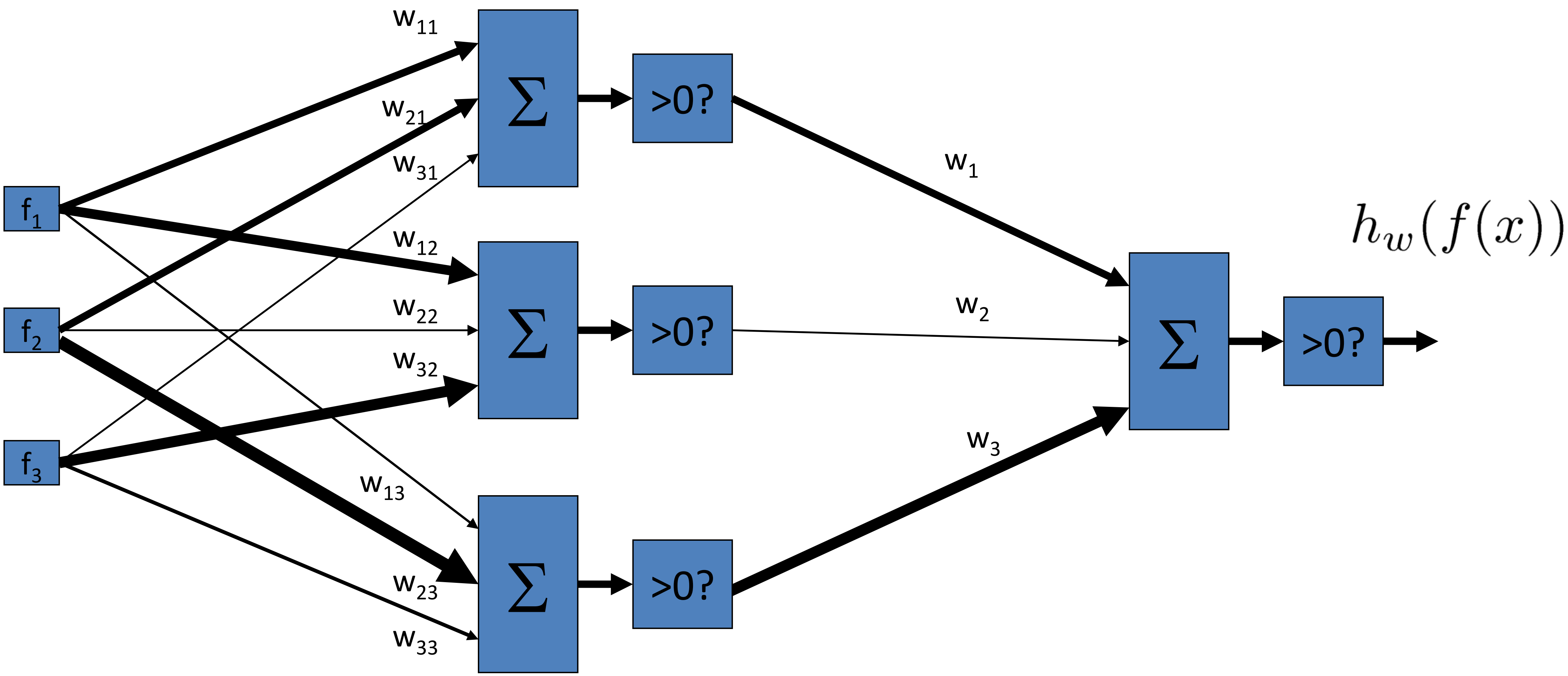


Neuronska Mreža

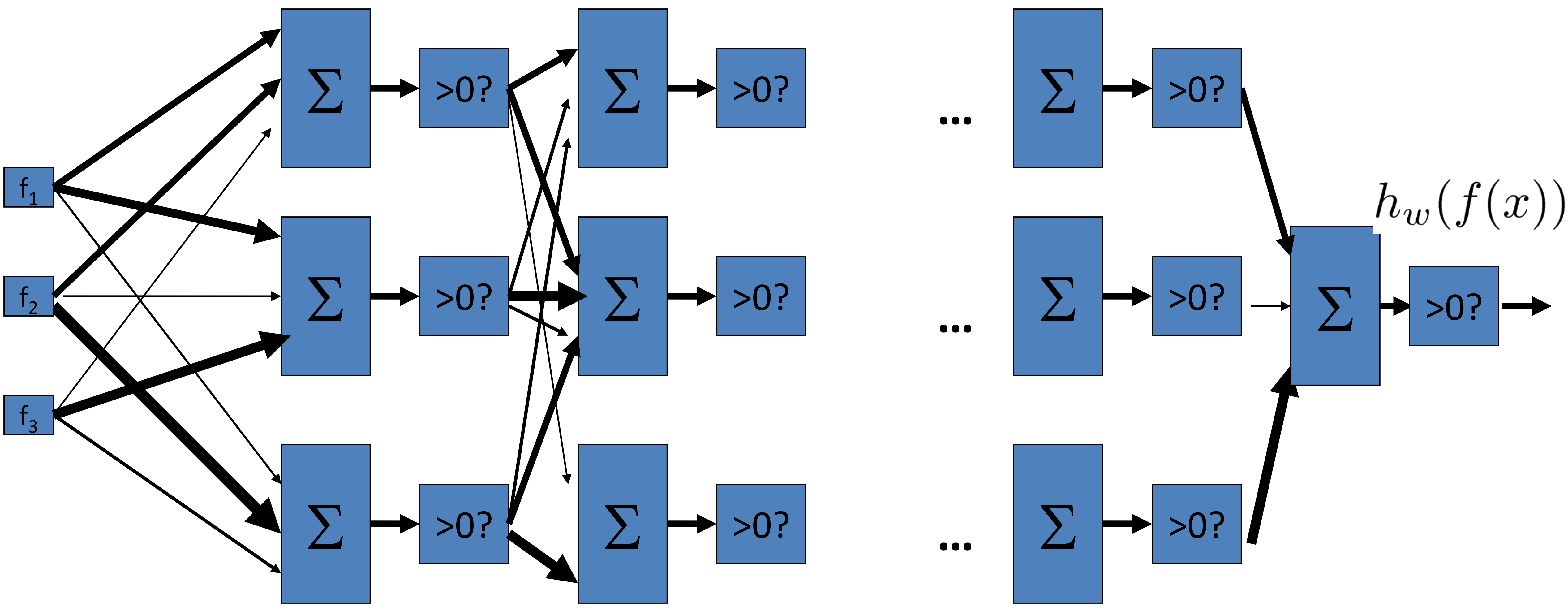
- Višeslojni perceptron je onda Neuronska Mreža!



Dvoslojni Perceptron

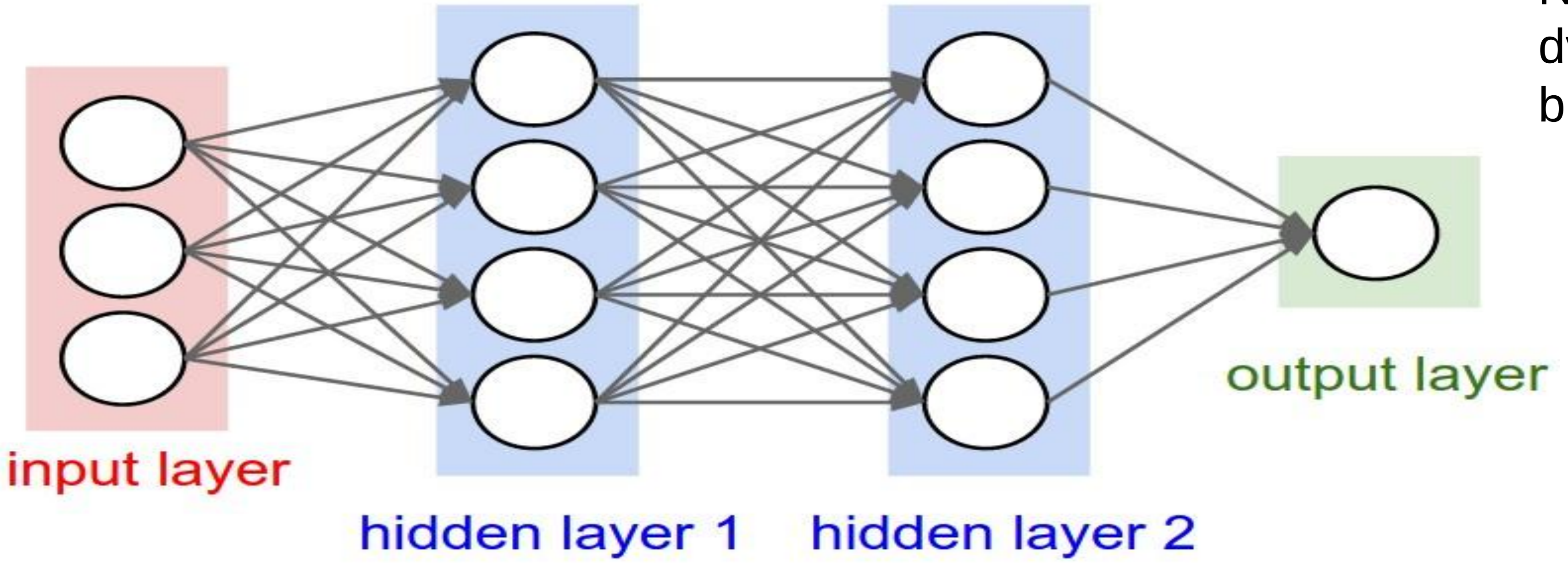


N-Slojni Perceptron



Neuronska Mreža - Organizacija

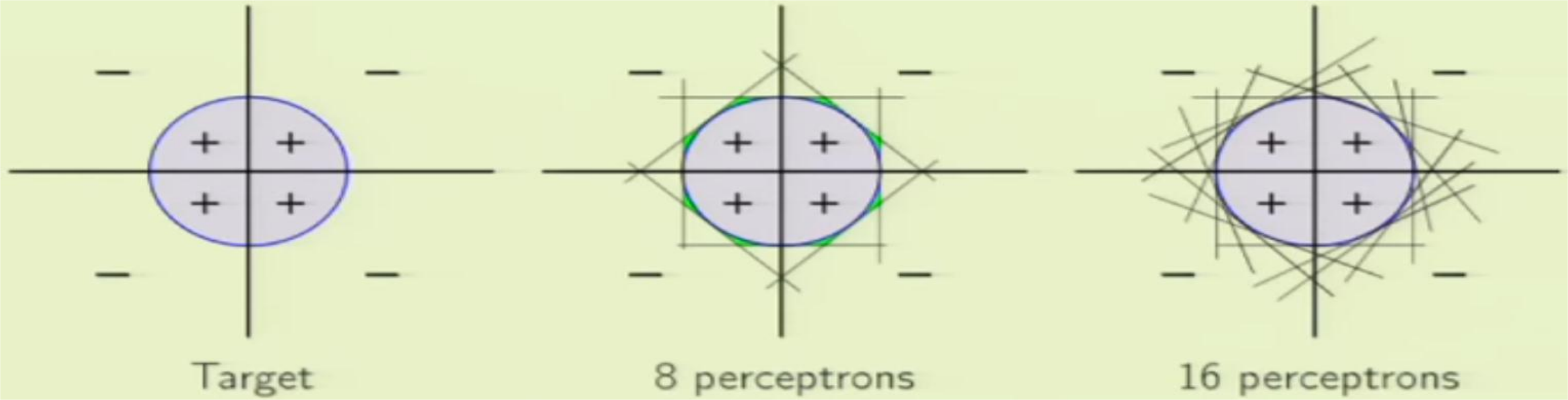
- Sloj obuhvata neurone i grane koje ulaze u njih
- Ovo je struktura sa propagacijom unapred (*feedforward*)
- Izlazi neurona su povezani isključivo na neurone sledećeg sloja
- Ovo nije jako restriktivno: ako možemo da implementiramo AND, OR i XOR možemo da implementiramo bilo šta!
- Dakle, možemo imati veoma sofisticirane granice odluke – samo nam treba dovoljno neurona koje ćemo kombinovati



Neuronska mreže sa tri sloja: dva skrivena i jedan izlazni (ne brojimo ulazni sloj)

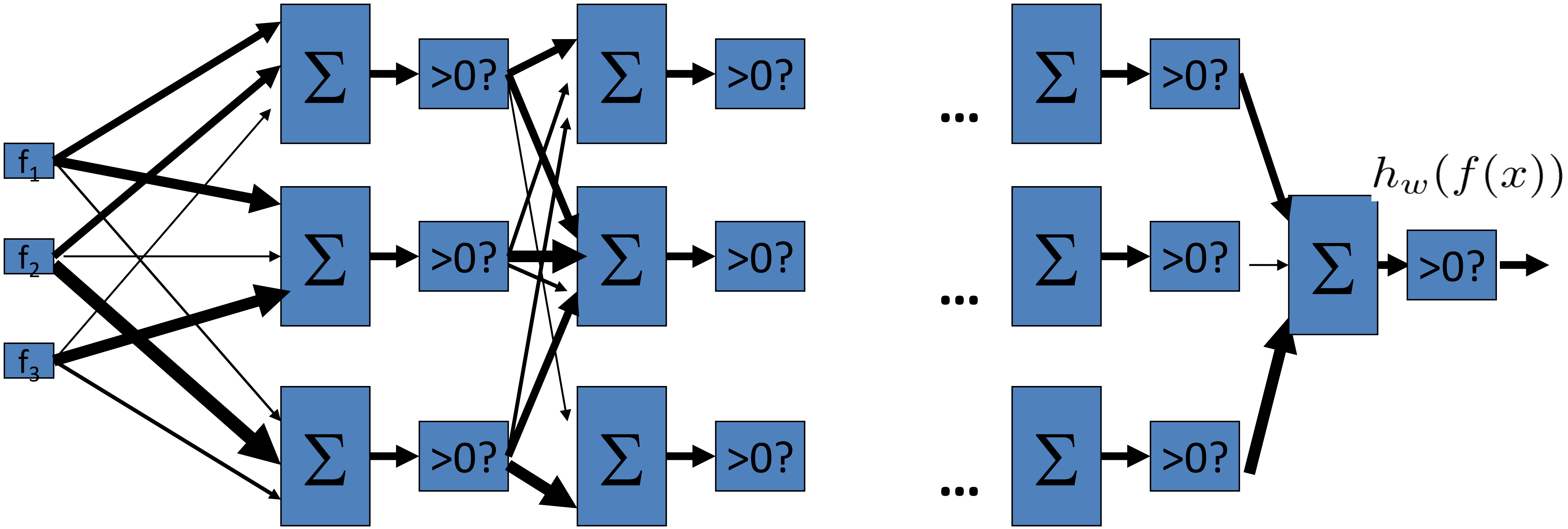
Neuronska Mreža – Moćan Model

- Možemo dokazati da MLP (Multy Layer Perceptron) sa dovoljno neurona može da aproksimira bilo koju funkciju
- Imamo izuzetno sofisticiran model, što je dobro, ali...
- Problem generalizacije – moramo pažljivo da prilagodimo sofisticiranost modela (arhitekturu – broj slojeva i neurona u svakom sloju) podacima kako ne bi došlo do overfittinga
- Problem optimizacije – velika količina parametara (težina) koja treba da se optimizuje, a treba nam optimalna kombinacija svih tih težina! Ovo je veoma težak optimizacioni problem.



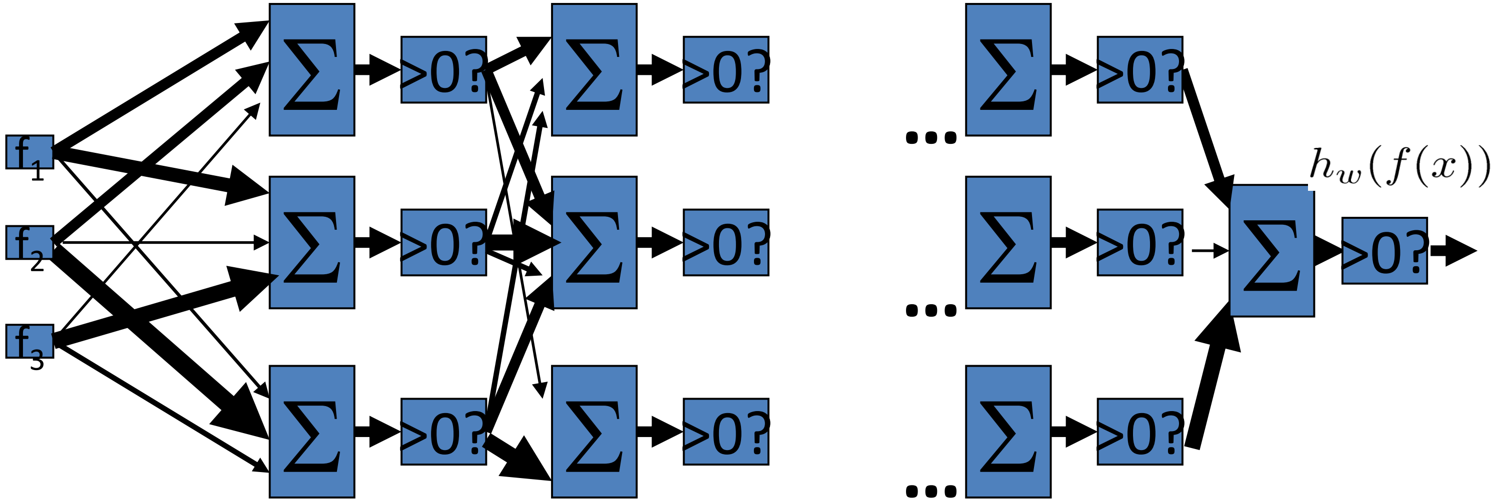
Kako obučiti Neuronsku Mrežu?

- Setimo se na koji smo način obučavali linearne klasfikatore.
- Koristili smo Gradijentni Spust. Da li ga možemo korisiti i sad?
- Možemo, ali postoji nekoliko ozbiljnih problema koje prvo moramo da rešimo.



Kako obučiti Neuronsku Mrežu?

- Problem 1: Funkcija greške. Višeslojni perceptron ima funkciju greške koja nije diferencijabilna i kod koje se izlaz naglo menja sa malom promenom ulaza.



$$l^{\text{acc}}(w) = \frac{1}{m} \sum_{i=1}^{m} \left(\text{sign}(w^\top f(x^{(i)})) == y^{(i)} \right)$$

Soft-Max Funkcija Greške

- Jedna alternativa za funkciju greške je softmax funkcija koju smo već pominjali
- Ako imamo dve klase:
- Skor za klasu y=1: $w^\top f(x)$ Skor za klasu y=-1: $-w^\top f(x)$
- Verovatnoće za klase:

$$p(y=1|f(x);w) = \frac{e^{w^\top f(x^{(i)})}}{e^{w^\top f(x)} + e^{-w^\top f(x)}}$$

$$p(y=-1|f(x);w) = \frac{e^{-w^\top f(x)}}{e^{w^\top f(x)} + e^{-w^\top f(x)}}$$

Soft-Max Funkcija Greške

- Ukupna verovatnoća za jedan vektor težina je:

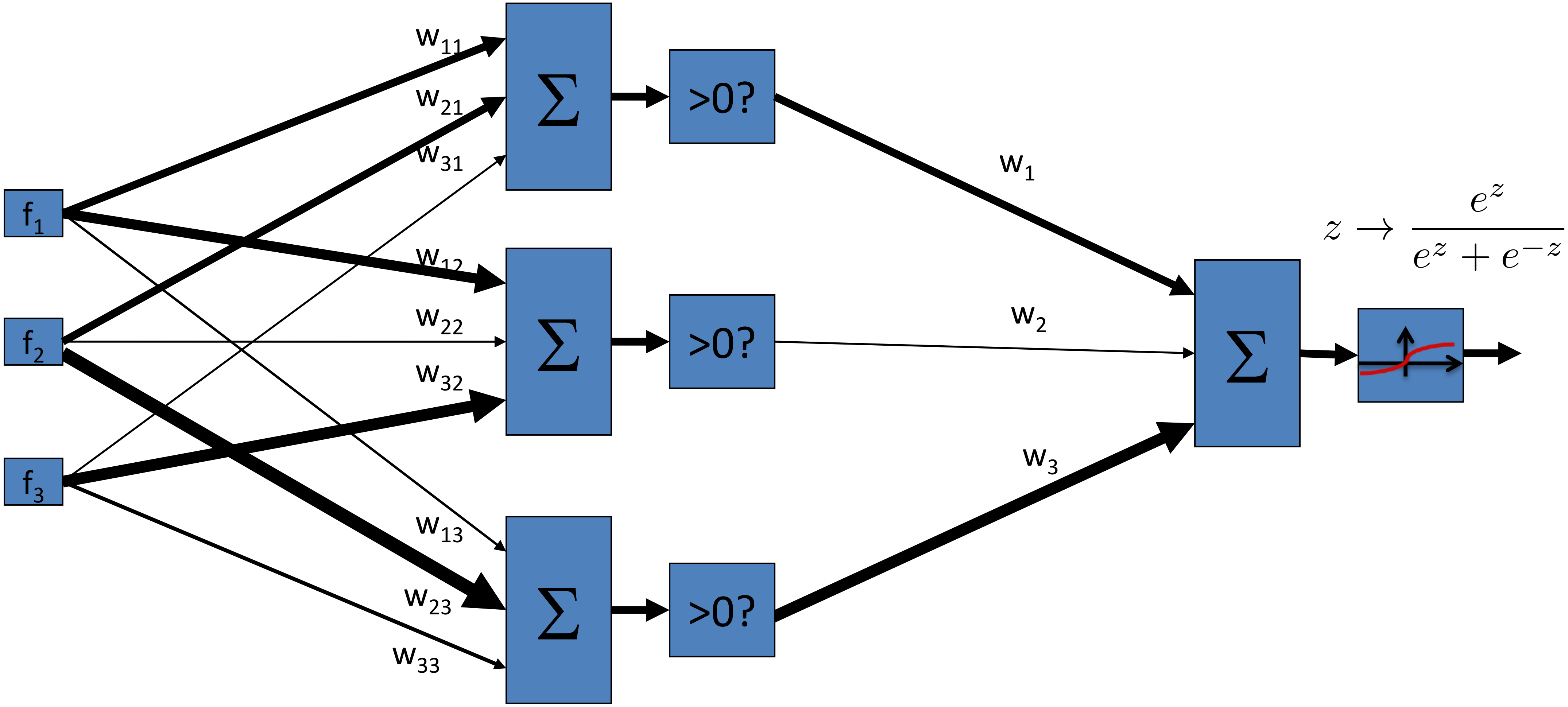
$$l(w) = \prod_{i=1}^{m} p(y = y^{(i)} | f(x^{(i)}); w)$$

- Dakle gradijentim spustom tražimo w takvo da je l(w) maksimalno
- U praksi se koristi logaritam od l(w) umesto l(w):

$$ll(w) = \sum_{i=1}^{m} \log p(y = y^{(i)} | f(x^{(i)}); w)$$

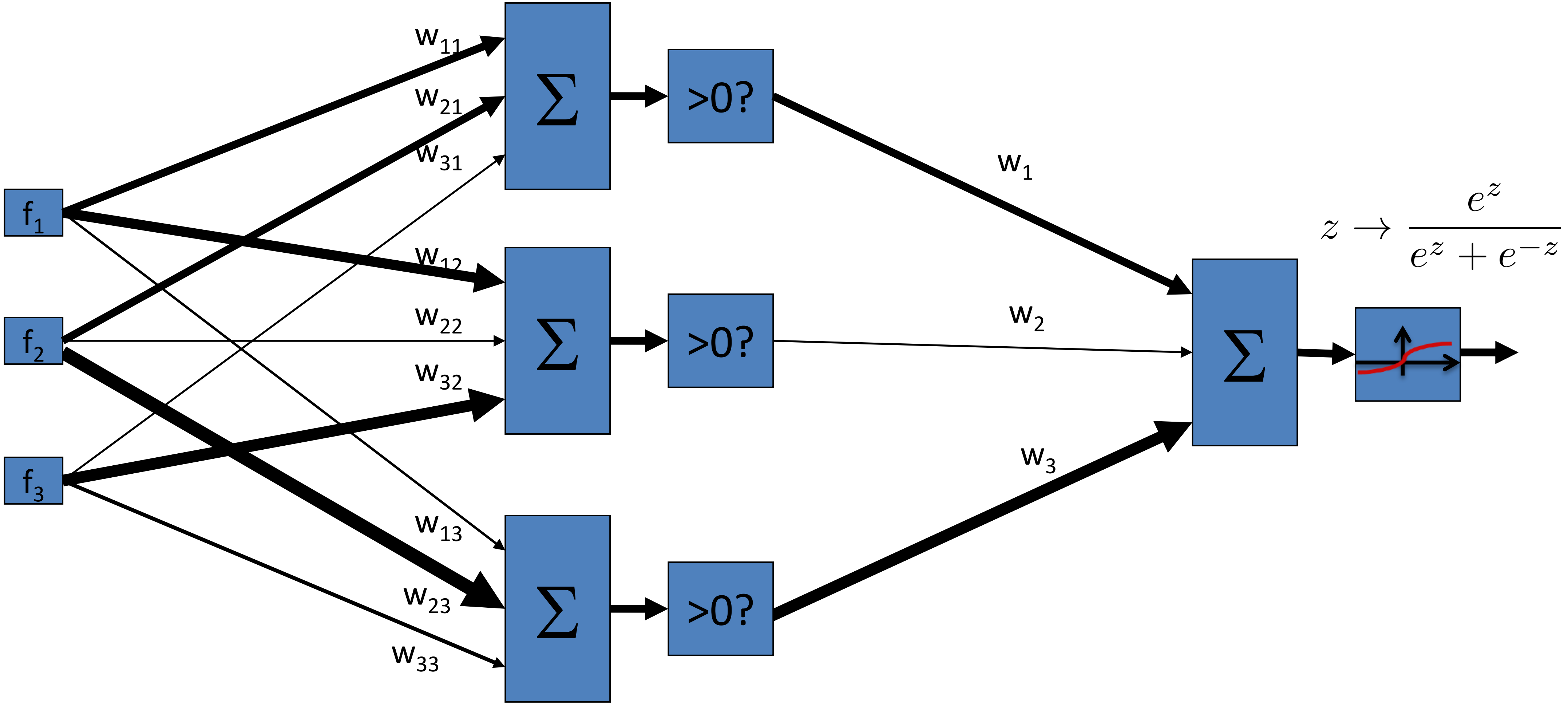
Dvoslojna Neuronska Mreža

- Sad imamo fukciju greške. Drugi problem je kako odrediti gradijent težina.



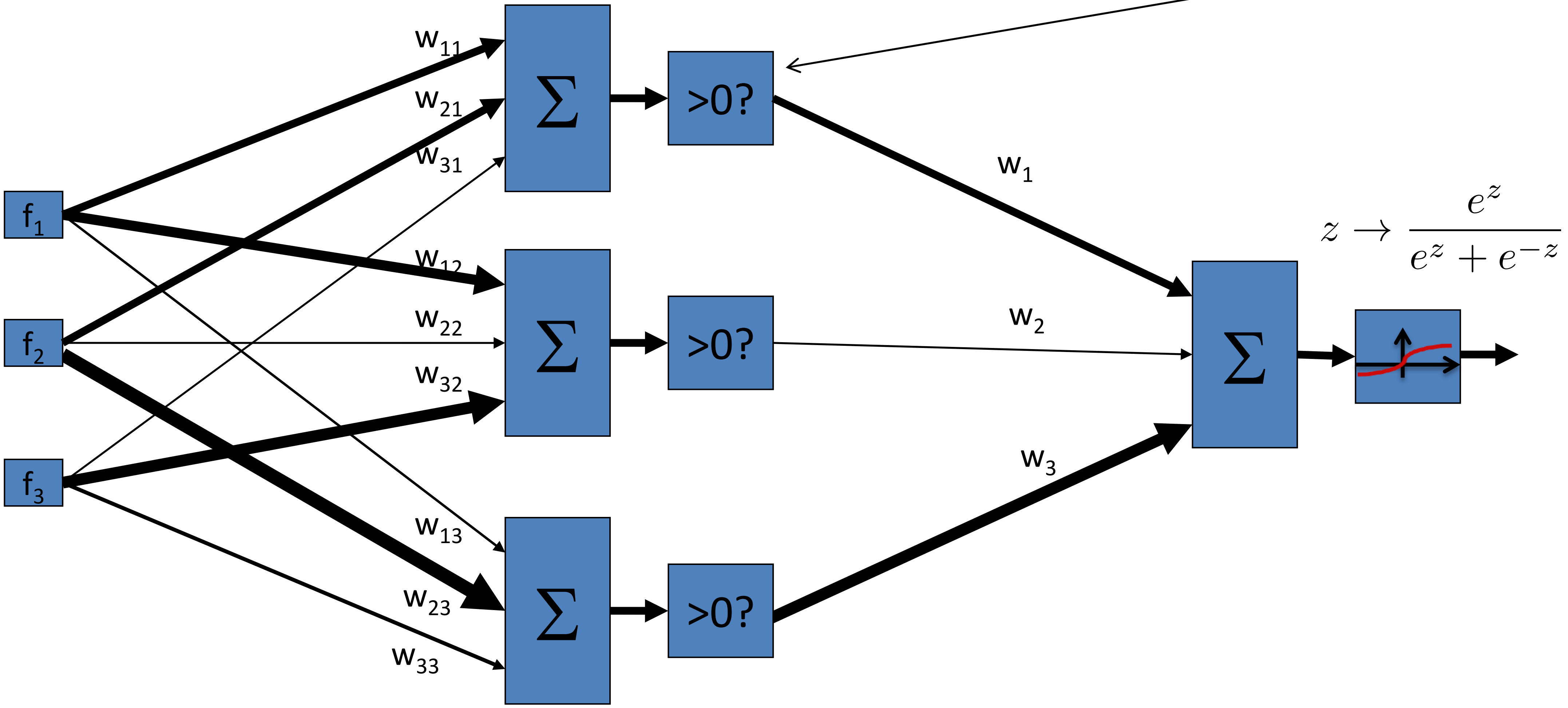
Dvoslojna Neuronska Mreža

- Lako je pomoću parcijalnih izvoda otkriti kako male promene w_1,w_2 i w_3 utiču na fukciju greške. Da li je to lako i za npr. w_{21}? Ne baš.



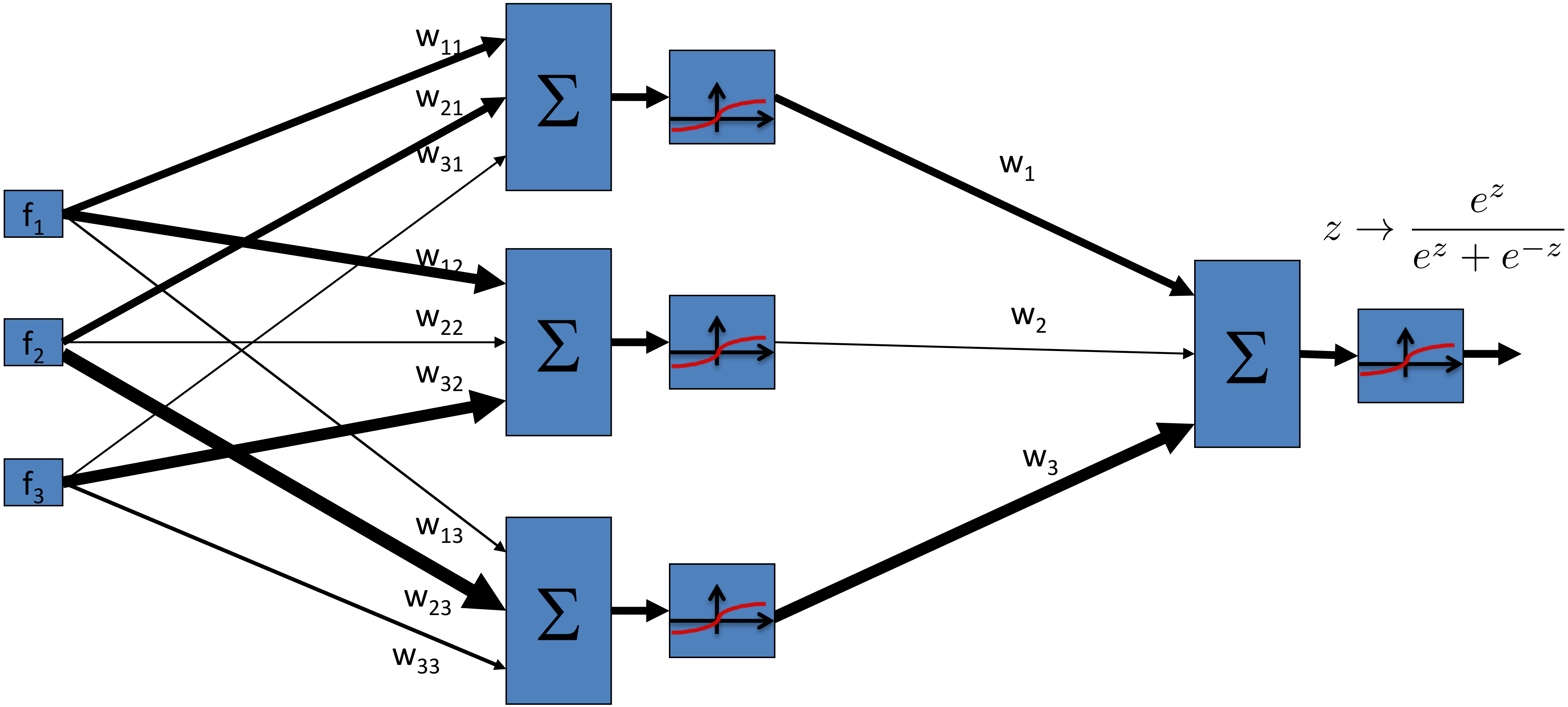
Dvoslojna Neuronska Mreža

- Da li je to lako i za npr. w_{21}? Ne baš.
- Prvi problem pri određivanju parcijalnih izvoda za w_{21} je što je ta težina uključena u funkciju aktivacije koja nije diferencijabilna (sign funkcija).



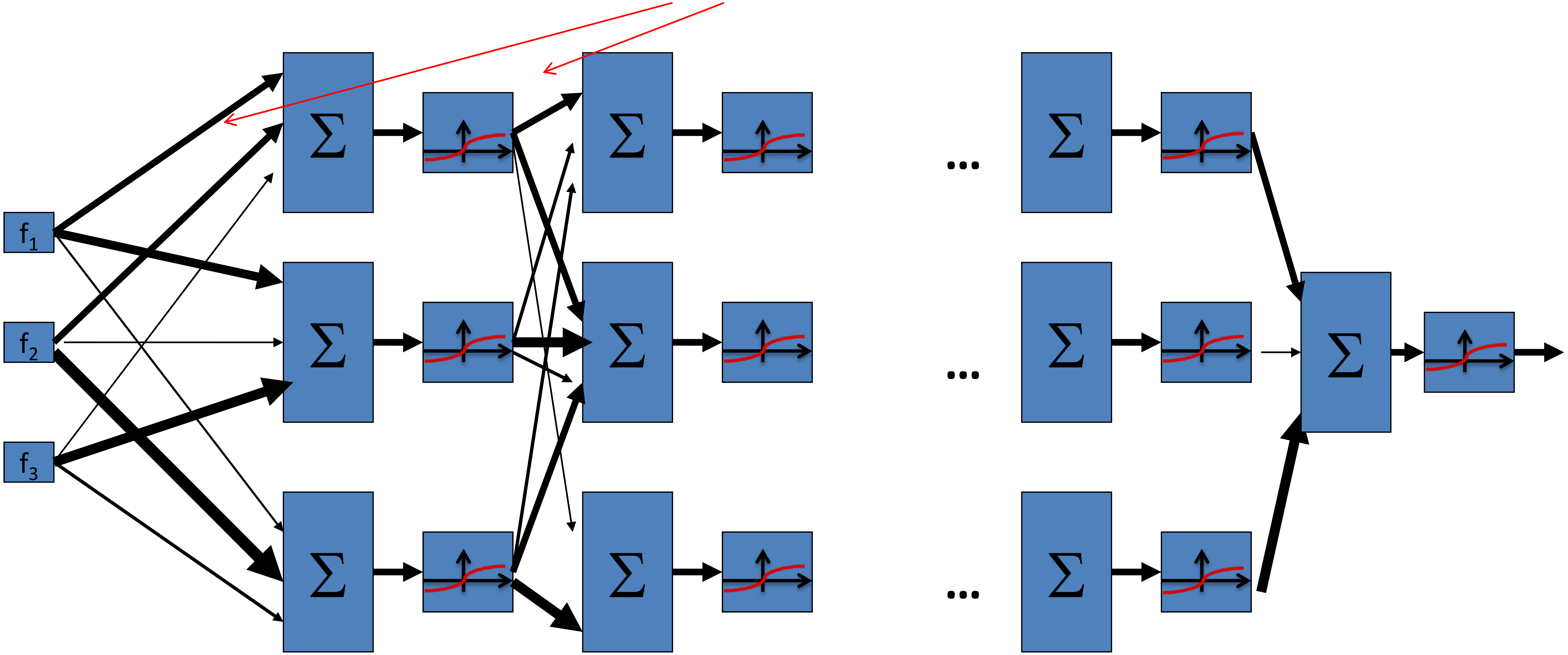
Dvoslojna Neuronska Mreža

- Prvi problem pri određivanju parcijalnih izvoda za w_{21} je što je ta težina uključena u funkciju aktivacije koja nije diferencijabilna (sign funkcija).
- Problem rešavamo uvođenjem diferencijabilnih funkcija aktivacije – više detalja na naredim predavanjima.



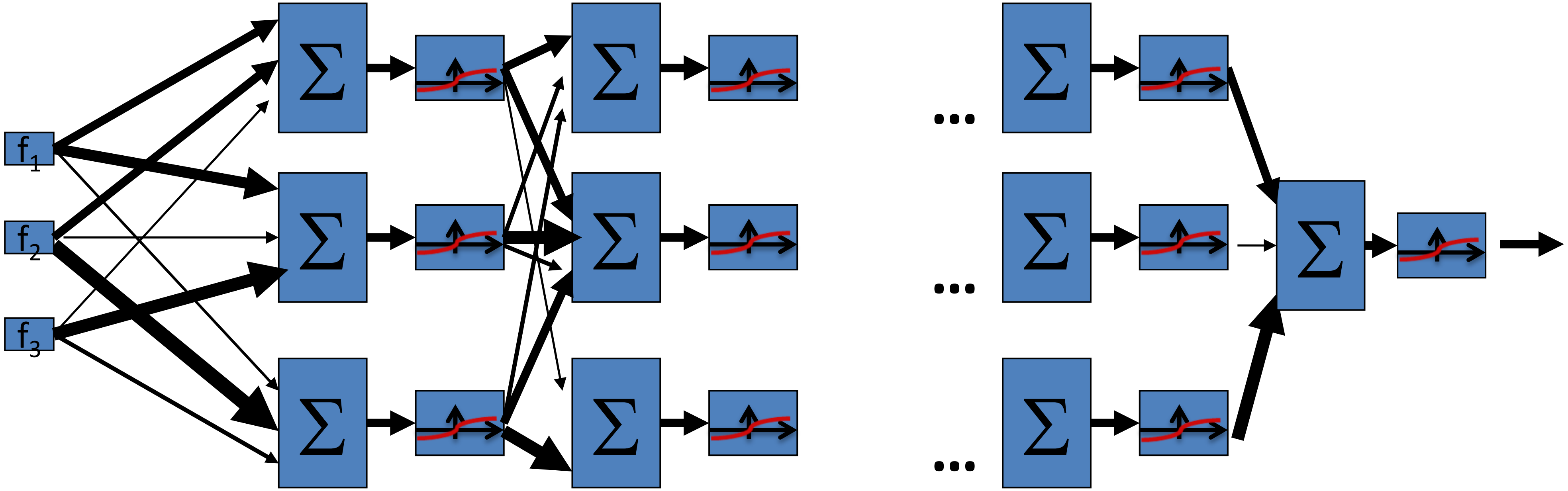
N-slojna Neuronska Mreža

- Da li je sad lako odrediti uticaj neke od ovih težina?



N-slojna Neuronska Mreža

- Da li je sad lako odrediti uticaj neke od ovih težina?
- Definitivno nije lako. Treba nam parcijalni izvod funkcije greške koja je na kraju, ali po težini koja je N slojeva ispod, gde N može biti 150 npr.
- Nećemo ga tražiti klasično, pomoću papira i olovke (iako je i to moguće).
- Posmatraćemo Neuronsku Mrežu kao Graf Izračunavanja i videti koliko je lako odrediti uticaj „dalekih težina“. O tome na sledećoj prezentaciji.



N-slojna Neuronska Mreža

- Pre nego što pređemo na sledeću prezentaciju, jedna napomena.
- Algoritam pomoću koga se izračunavaju uticaji (formalnije gradijenti) svih težina u mreži zove se:
  - Probagacija Unazad ili *Backpropagation*
- To je fundamentalni algoritam za obučavajnje neuronskih mreža i morate ga razumeti.
- Ako ga ne razumete, nećete moći da se izborite sa svim situacijama do koji može doći kad obučavate npr. konvolutivnu mrežu od 150 slojeva.