ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

# Невронски мрежи

# Невронски мрежи

- Два погледа на невронските мрежи:
  - од пресметковен (компјутерски) аспект – вештачки невронски мрежи – метод за претстава на функции со помош на едноставни аритметички пресметковни елементи (и нивна обука од примероци)
  - од биолошки аспект - математички модел на работата на мозокот
- Неврони – клетки кои ја изведуваат обработката на информациите во мозокот
- Мрежа од меѓусебно поврзани неврони – невронска мрежа

# Што се тоа „вештачки невронски мрежи“ (Artificial Neural Networks)?

- Парадигма за обработка на информации инспирирана од начинот на кој природните мозоци ја обработуваат информацијата, составени од густо меѓуповрзани паралелни структури
- Вид на повеќепроцесорски системи со:
  - Едноставни процесорски елементи
  - Висок степен на меѓусебна поврзаност
  - Едноставни пораки со броеви
  - Прилагодлива интеракција помеѓу елементите

# Мозок

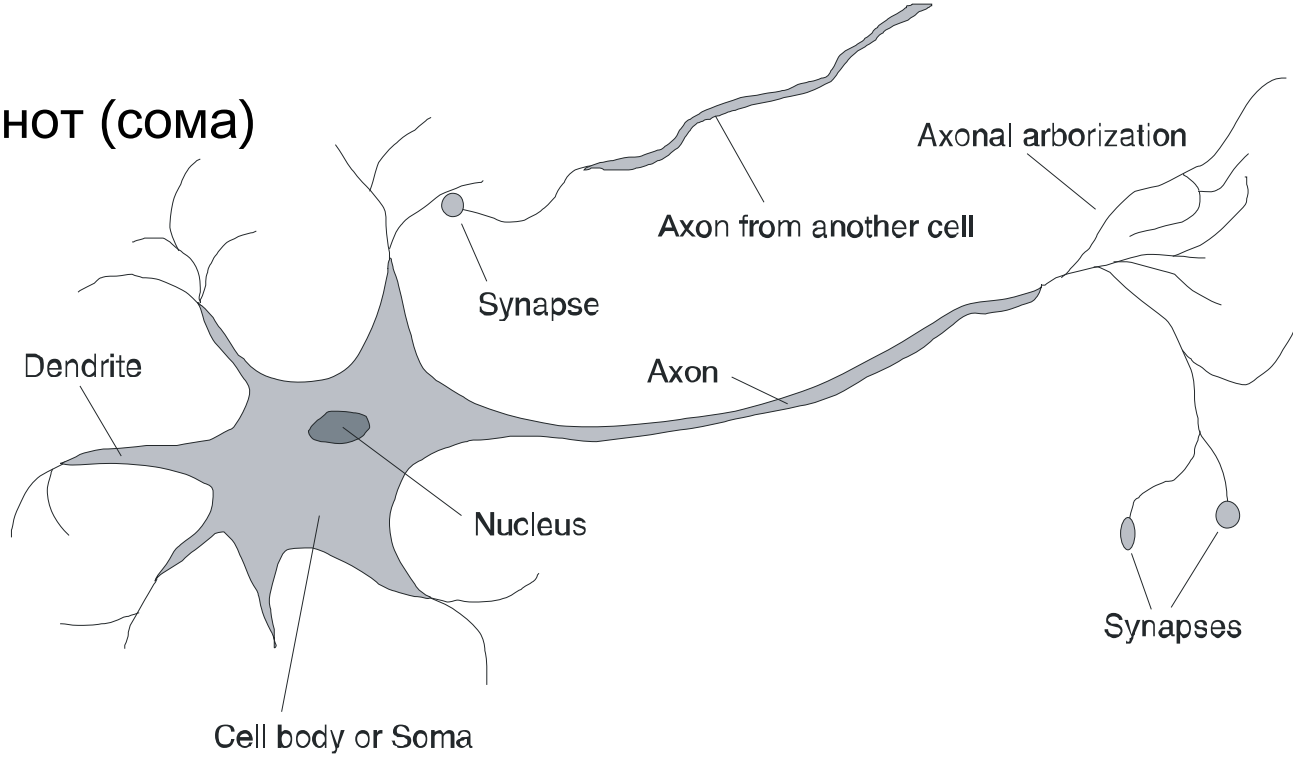
- Како работи мозокот (мистерија?)
- 335 пр.н.е. Аристотел - „Од сите животни, човекот има најголем мозок во однос на неговата големина“
- Неврон – фундаментална функционална единица на сите нервни ткива

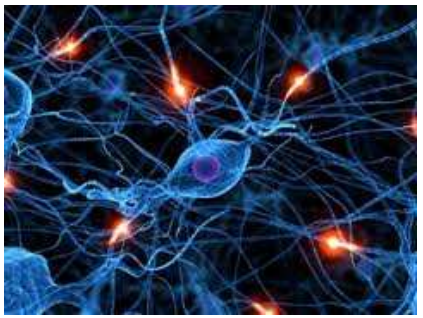
# Работа на невронот

- Сложен електрохемиски процес
- Акциски потенцијал – ниво на активација
- Синапси - ексцитациски (поттикнувачки) / инхибиторни (спречувачки)
- Пластичност

# Неврон

- Структура:
  - Тело на невронот (сома)
  - Аксон
  - Дендрити
  - Синапси

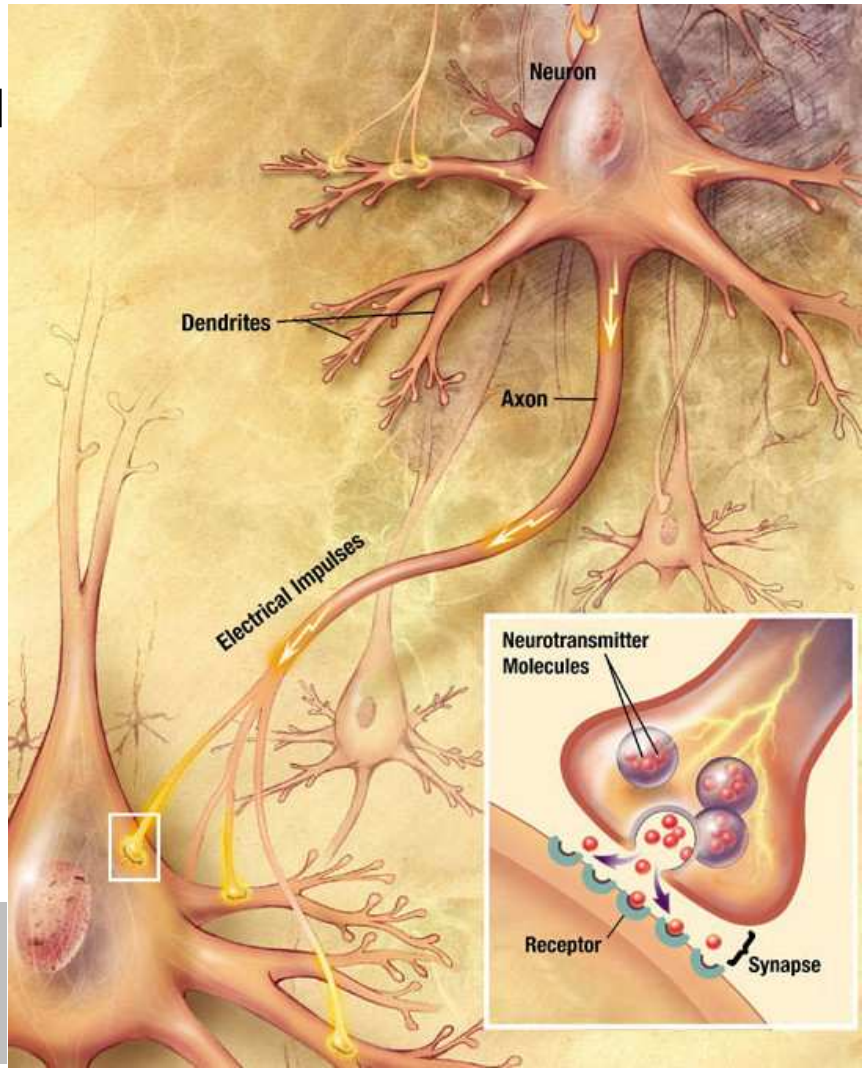


# Работа на невронот - синапси

- Сумарен постсинаптички потенцијал (СПП)
- Синаптичко влијание
  - екситациско
  - инхибиторно
- Пластичност

Синапсите се основата за меморирањето и учењето



# Споредба

- Кај мозокот сите неврони работат симултано

| | Computer | Human brain |
|---|---|---|
| Computational units | 62 ~~1~~ CPU, ~~$10^8$~~ $10^9$ gates | $10^{11}$ neurons |
| Storage Units | $10^{10}$ bits RAM<br>$10^{11}$ bits disk | $10^{11}$ neurons<br>$10^{14}$ synapses |
| Cycle time | $10^{-9}$ sec | $10^{-3}$ sec |
| Bandwidth | $10^{10}$ bits/sec | $10^{14}$ bits/sec |
| Memory update/sec | $10^9$ | $10^{14}$ |

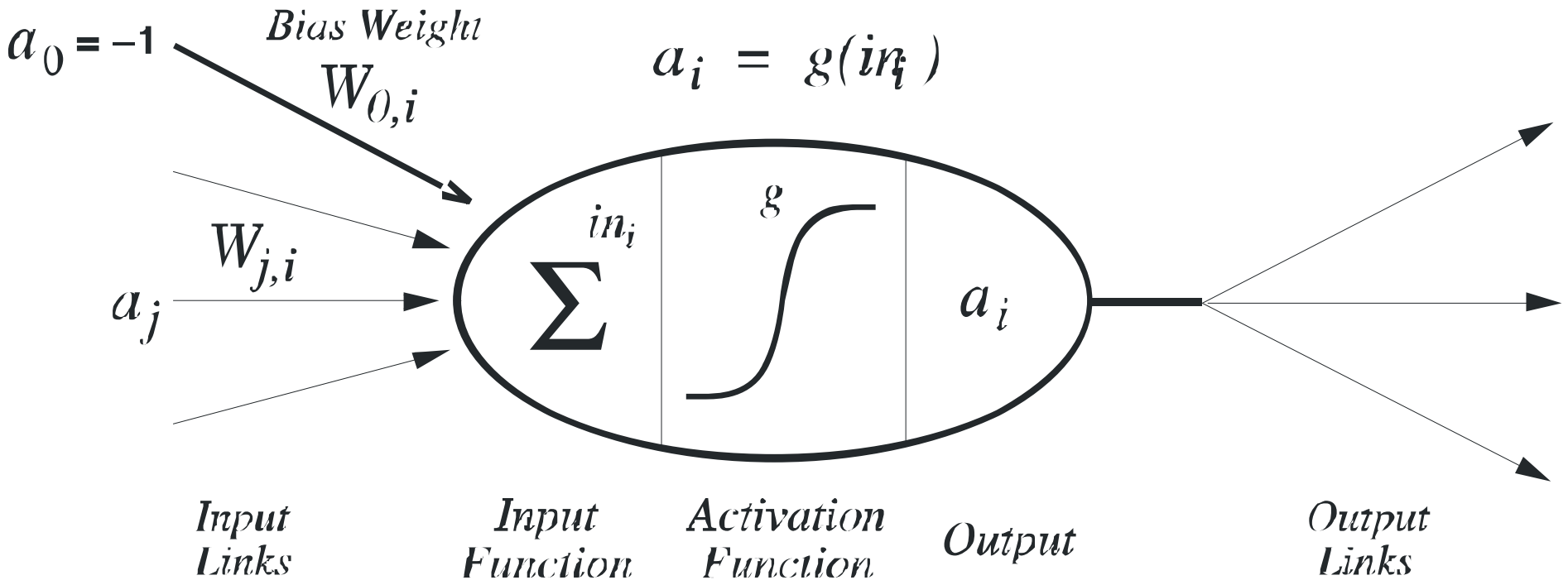
- брз
- точен
- (претежно) секвенцијален
- (брзо и точно) пресметување сложени математички изрази

- бавен
- fault-tolerant
- паралелен
- брзо препознавање на комплексни слики (сцени)

# Невронски мрежи

- Неврони (јазли), врски, тежини, активациско ниво (праг)
- За да се направи невронска мрежа која ќе решава одредена задача потребно е:
  - да се одреди бројот на невроните
  - нивниот тип
  - како ќе бидат поврзани
  - да се иницијализираат тежините и со некој алгоритам да се обучи мрежата претставувајќи и познати примероци
  - да се одлучи како ќе биде претставен проблемот (влез / излез)

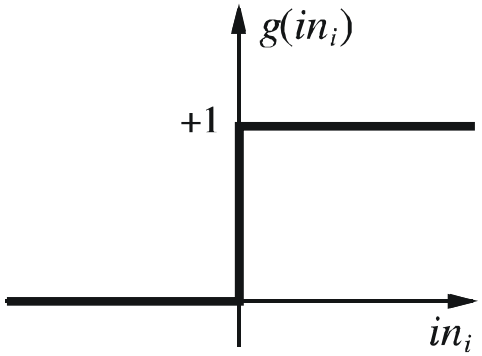
# Вештачки неврон

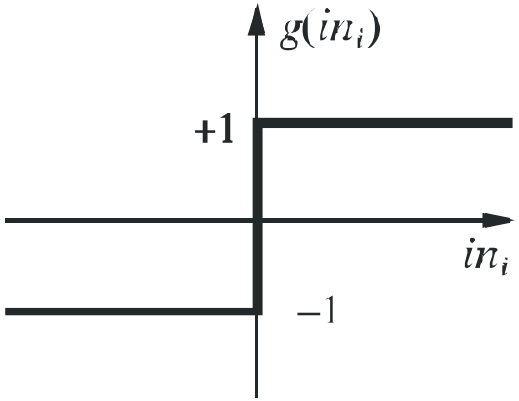


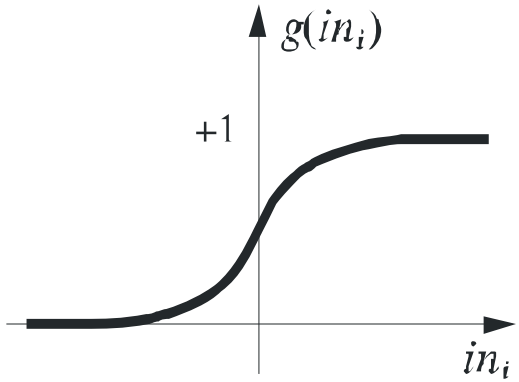

$$in_i = \sum_j W_{j,i} a_j = \mathbf{W}_i \cdot \mathbf{a}_i$$

# Активациска функција

- Активација $$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$
- Различни функции $g$ – различни модели на неврони



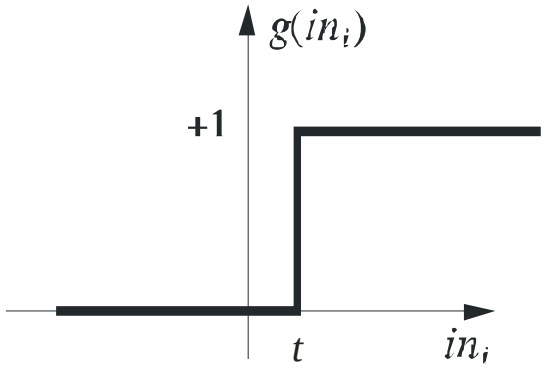

$$step_t(x) = \begin{cases} 1, \text{ if } x \geq t \\ 0, \text{ if } x < t \end{cases}$$

$$sign(x) = \begin{cases} +1, \text{ if } x > t \\ -1, \text{ if } x < t \end{cases}$$

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

# Праг на активација

- Практично: Замена на прагот со дополнителен влез со одредена тежина

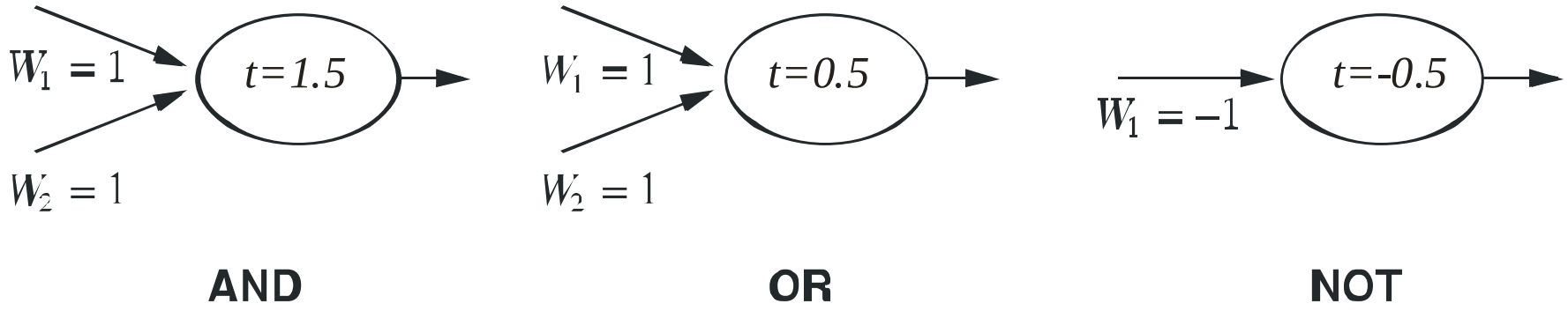


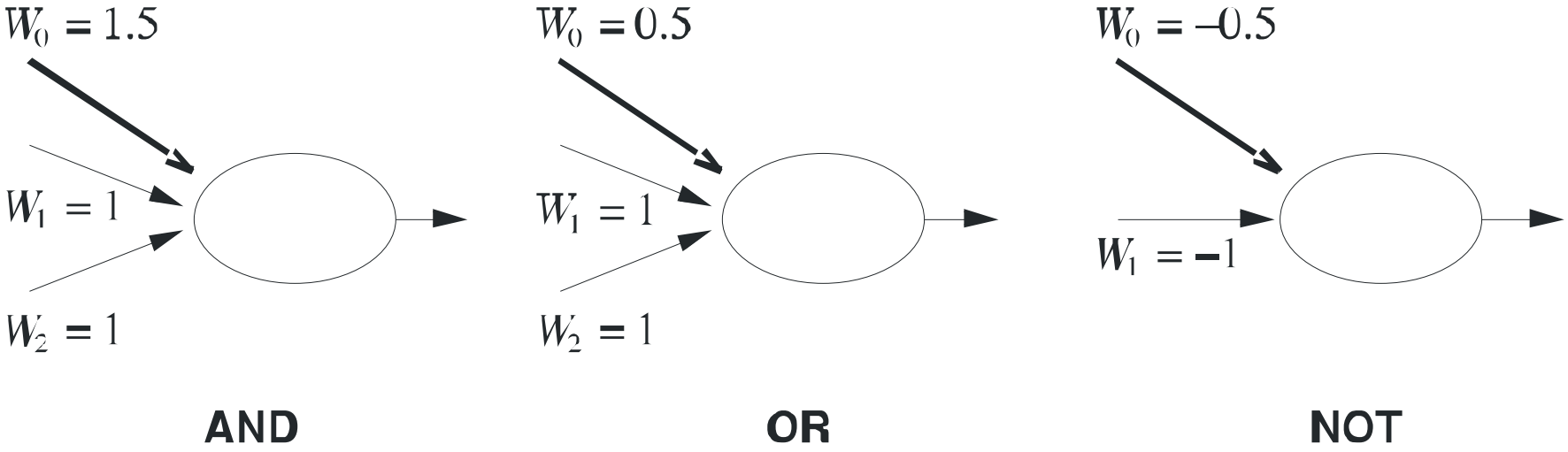

$$a_i = step_t\left(\sum_{j=1}^{n} W_{j,i} a_j\right) = step_0\left(\sum_{j=0}^{n} W_{j,i} a_j\right)$$

каде $W_{0,I} = t$ и $a_0 = -1$

# Логички функции со невроните

- Модели на неврони кои ги реализираат основните логички функции



# Наједноставно учење кај НМ

- Хебово учење
  - Во 1949, Доналд Хеб напишал: „Кога аксонот на клетката А е доволно блиску до клетката Б за да може да ја ексцитира (возбуди) и повторливо и истрајно учествува во нејзиното палење (активирање), некои процеси на растење и метаболитски промени се случуваат во едната или во двете клетки, така што ефикасноста на А во возбудувањето на Б е зголемена“

# Хебово учење (Hebbian Learning)

- Психолозите и когнитивните научници ја генерализираа оваа идеа така што кога два перцепти повторливо се набљудуваат заедно, во меморијата се формира асоцијација помеѓу нив, така што едниот перцепт може да го повика (invoke) другиот.

# Формализација на Хебовото учење

- Изградба на асоцијации заради истовремено појавување (Association-Building by Co-occurrence)
- Формализирано како вид на ненадгледувано учење кај НМ
- $\Delta w_{ij} = \eta \cdot a_i \cdot a_j$
  - промена на тежината помеѓу невроните $i$ и $j$ со степен на учење (learning rate) $\eta$

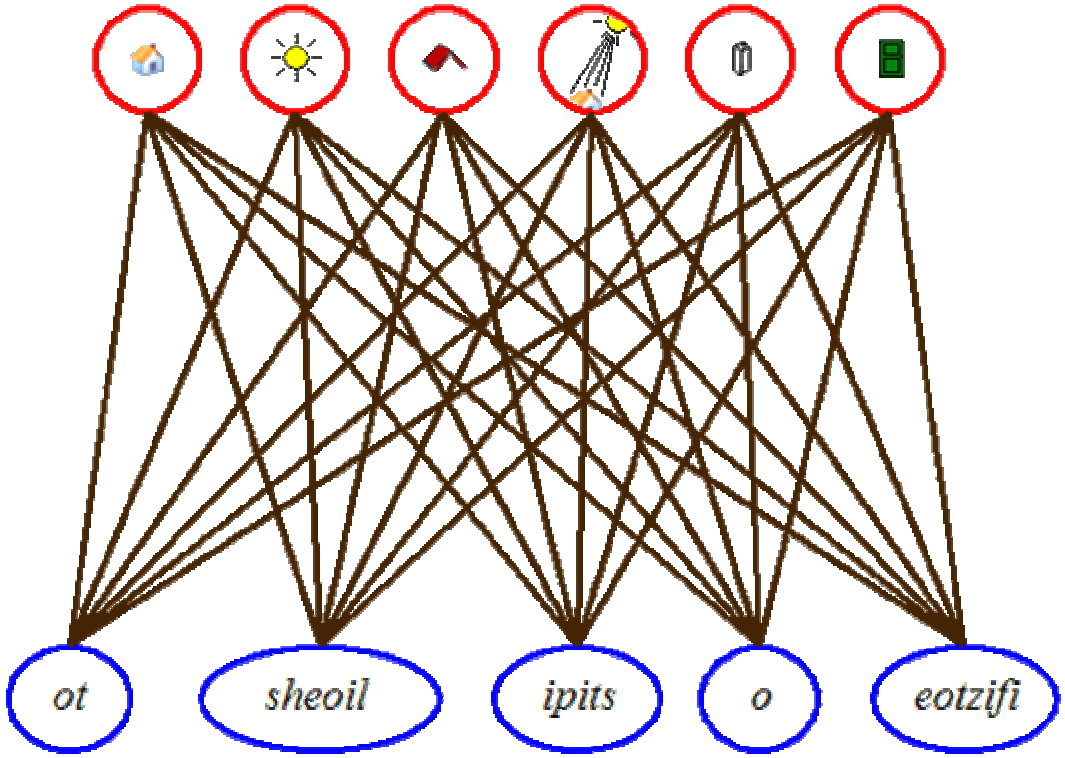
# Пример за Хебово учење (1)*

- Замислете дека треба да научите некој странски јазик (како што при средбата на Индијанците и Европејците се учеле странските јазици, на почетокот).
- Визуелниот и ауидиторниот влез се даваат во парови на слика со реченица која веројатно ја објаснува таа слика

*o sheoil eotzifi ot ipits*

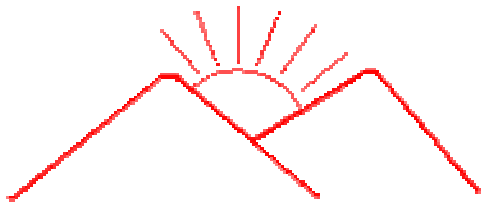
# Пример за Хебово учење (2)*

- Формирање на асоцијации помеѓу секој визуелен перцепт и секој лингвистички симбол (збор)



# Пример за Хебово учење (3)*

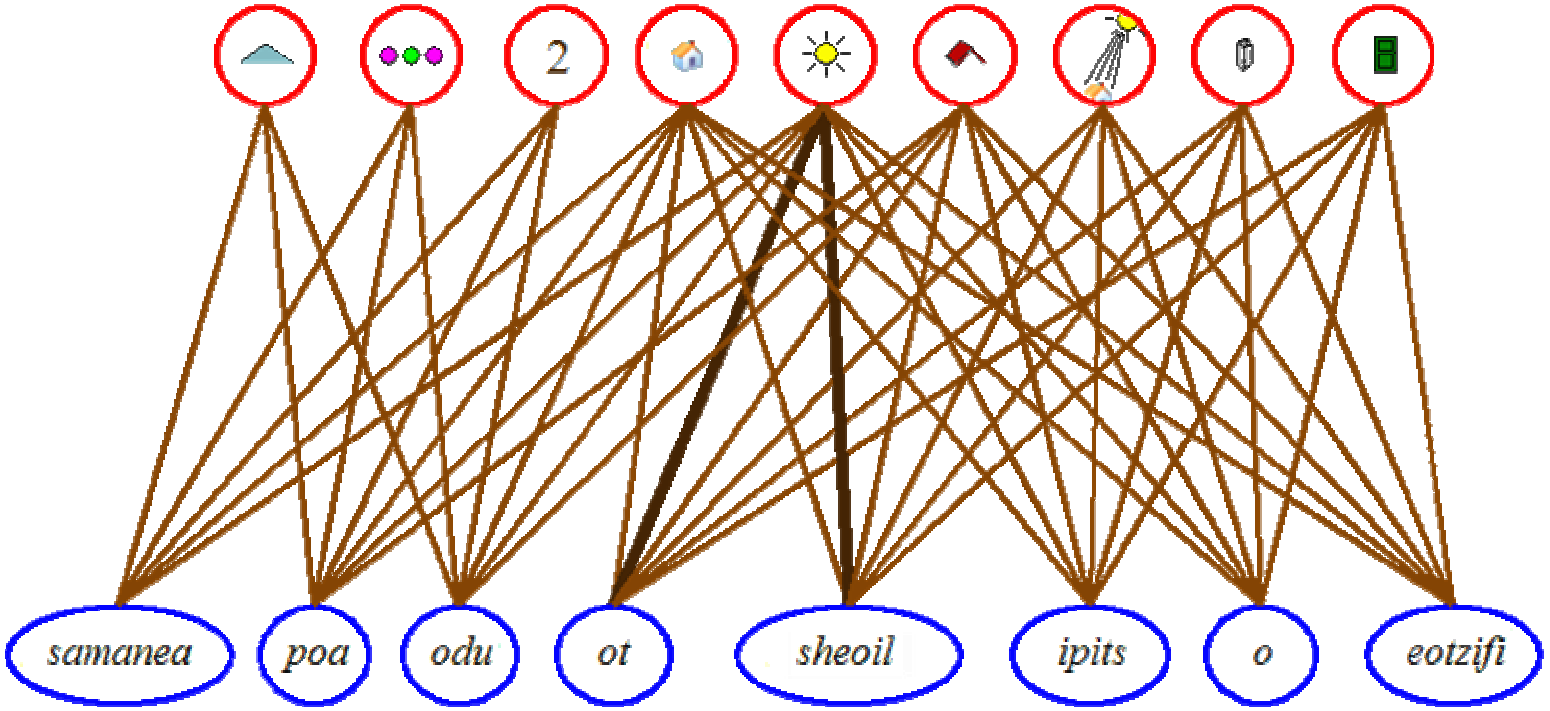
- Сега добивате нов пар на слика и реченица



*o sheoil ot eotzifi samanea poa odu onbau*

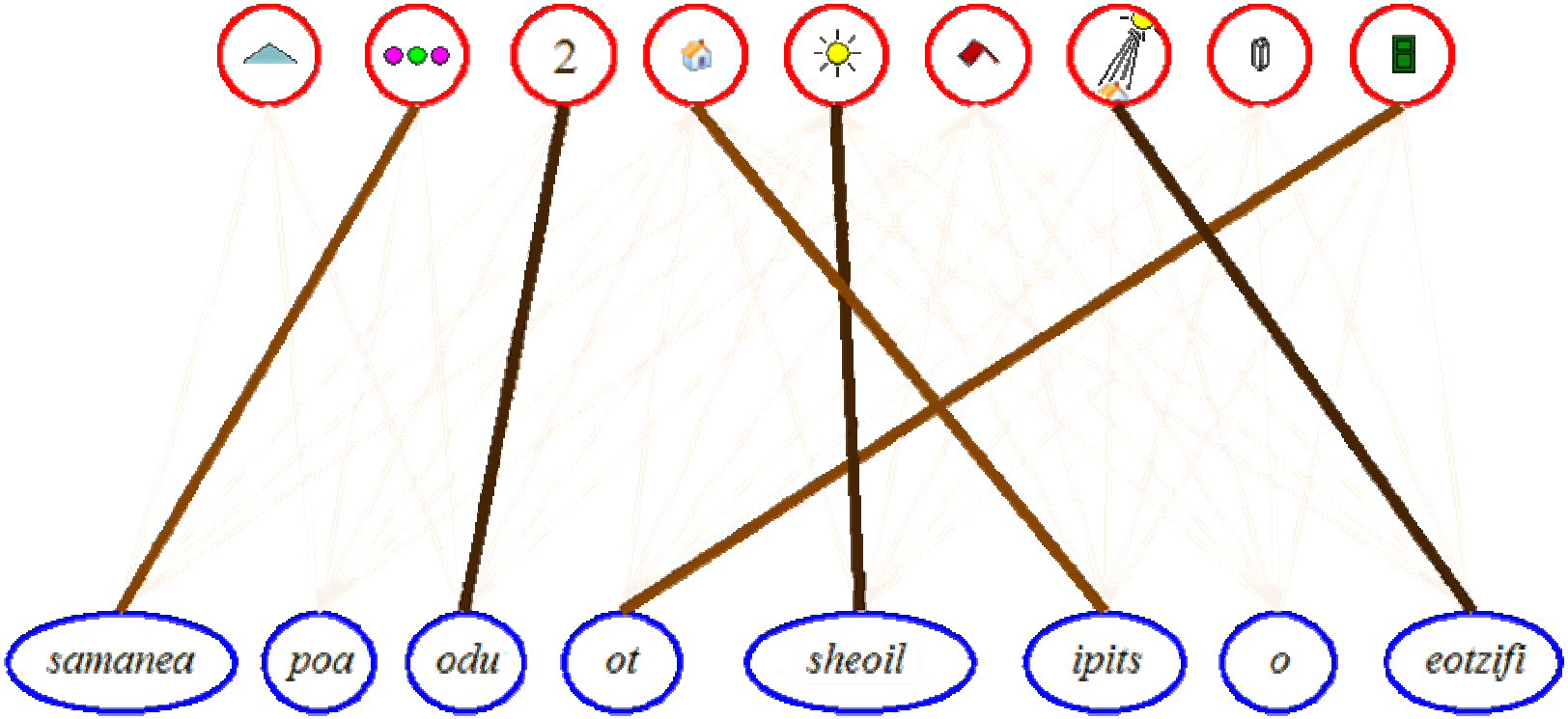
# Пример за Хебово учење (4)*

- Сега сликите и зборовите се додаваат како унија во соодветните множества и тие асоцијации каде повторно се појавија истите слики и зборови се засилуваат за разлика од другите кои се заслабнуваат

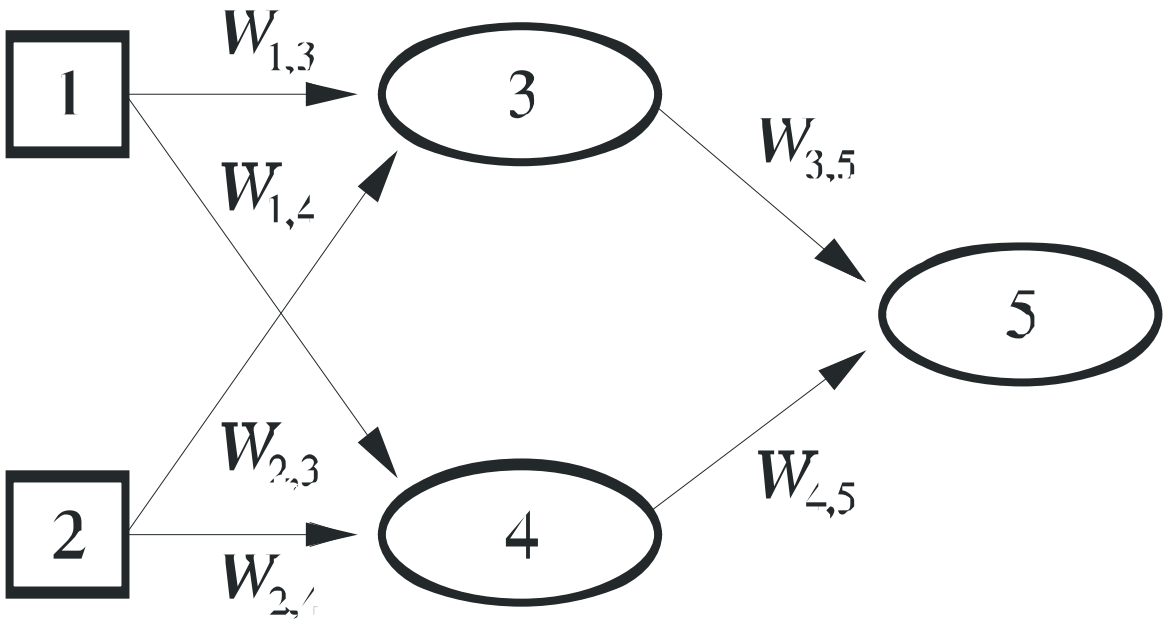


# Пример за Хебово учење (5)*

- И така, со тек на време, со прикажување на многу парови на слики и реченици, ќе се издвојат (правилните) асоцијации помеѓу сликите и зборовите
- Колку ќе бидат правилни асоцијациите зависи од учителот



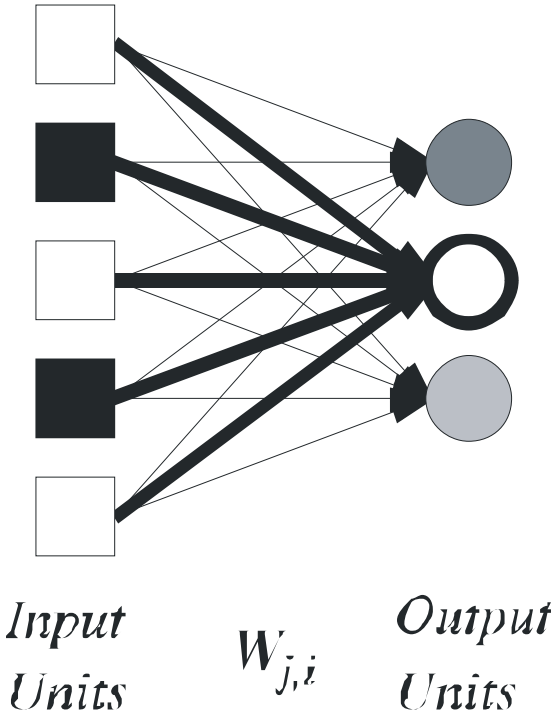
# Двослојна невронска мрежа




$$a_5 = g(W_{3,5}a_3 + W_{4,5}a_4)$$
$$= g(W_{3,5}g(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1 + W_{2,4}a_2))$$

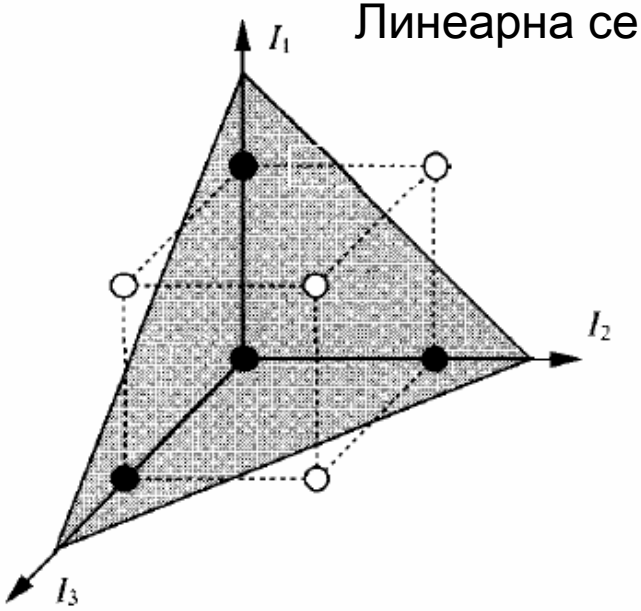
# Перцептрон

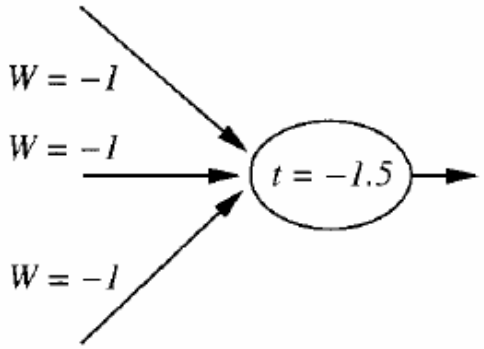
- Перцептрон = feed-forward невронска мрежа со едно ниво
- Што може да репрезентираат перцептроните?
  - majority може, XOR не може
- Перцептронот може да репрезентира само линеарно сепарабилни функции!

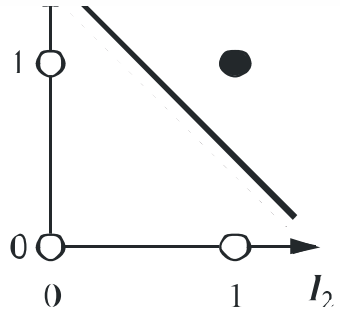


# Ограничувања на перцептронот

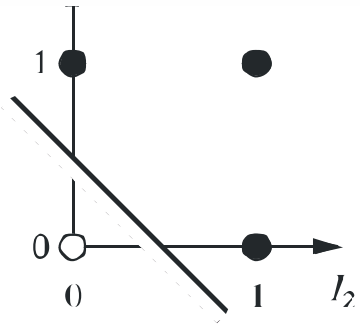
Линеарна сепарабилност (делливост) во хипер-рамнина



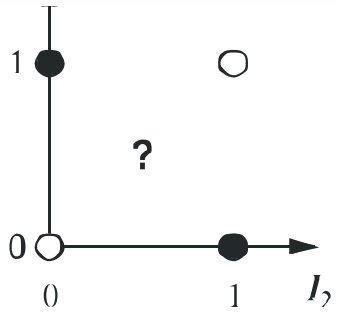




(a) $I_1$ **and** $I_2$




(b) $I_1$ **or** $I_2$




(c) $I_1$ **xor** $I_2$

# Обука на перцептронот

- Постои алгоритам кој може да научи било која линеарно сепарабилна функција ако му се дадат доволно примероци за обука

1. Иницијализација на тежините на случајни вредности
2. Промена на овие вредности со цел да се направат конзистентни со примерите – итеративен процес на мали промени на тежините со кои би се намалила разликата помеѓу добиениот и бараниот излез за даден примерок - конвергенција

# Правило за промена на тежините

- Епоха – промена на сите тежини за сите примероци
- Правило за промена на тежините

$$Err = T - O$$

- Секој влез придонесува $W_j I_j$ на вкупниот излез

$$W_j \leftarrow W_j + \alpha \cdot I_j \cdot Err$$

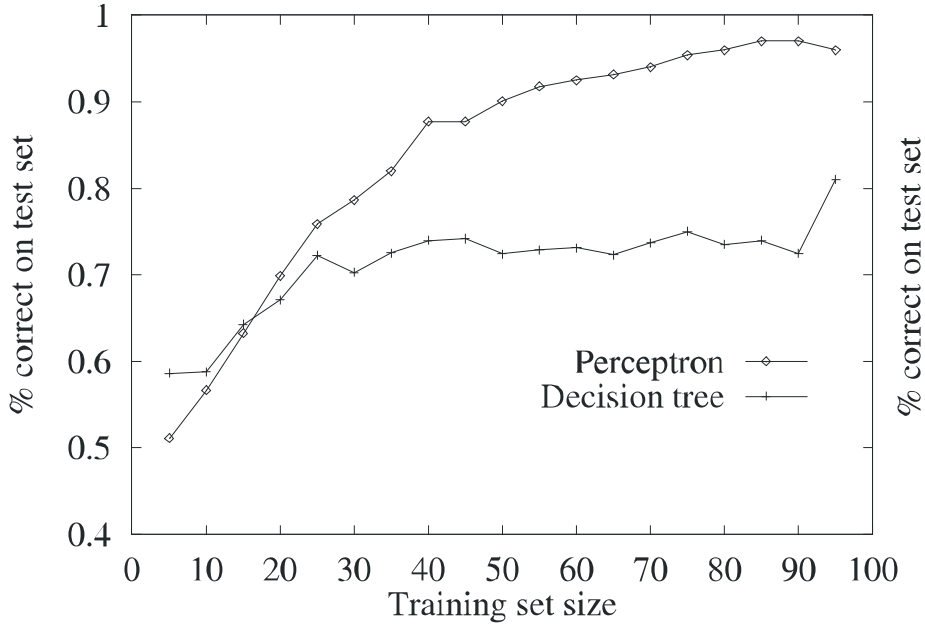
- $\alpha$ - стапка на обука (learning rate)

# Generic NN learning – општо учење кај НМ*

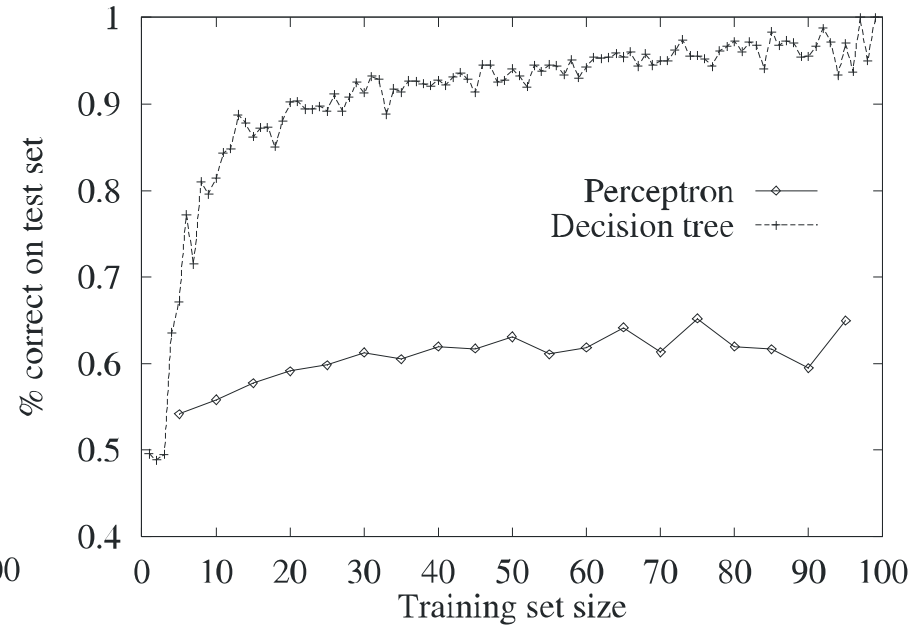
**function** NEURAL-NETWORK-LEARNING(*examples*) **returns** *network*

*network* ← a network with randomly assigned weights
**repeat**
**for each** *e* **in** *examples* **do**
O ← NEURAL-NETWORK-OUTPUT(*network*, *e*)
T ← the observed output values from *e*
update the weights in *network* based on *e*, O, and T
end
**until** all examples correctly predicted or stopping criterion is reached
**return** *network*

# Обука на НМ и дрвата на одлука




majority (11 bits)




restaurant

# Повеќеслојни невронски мрежи

- Multilayer feed-forward neural networks (Rosenblatt 1950s)
- Проблем – обуката
- Back-propagation алгоритам (1969, 1980)

# Повеќеслојни невронски мрежи

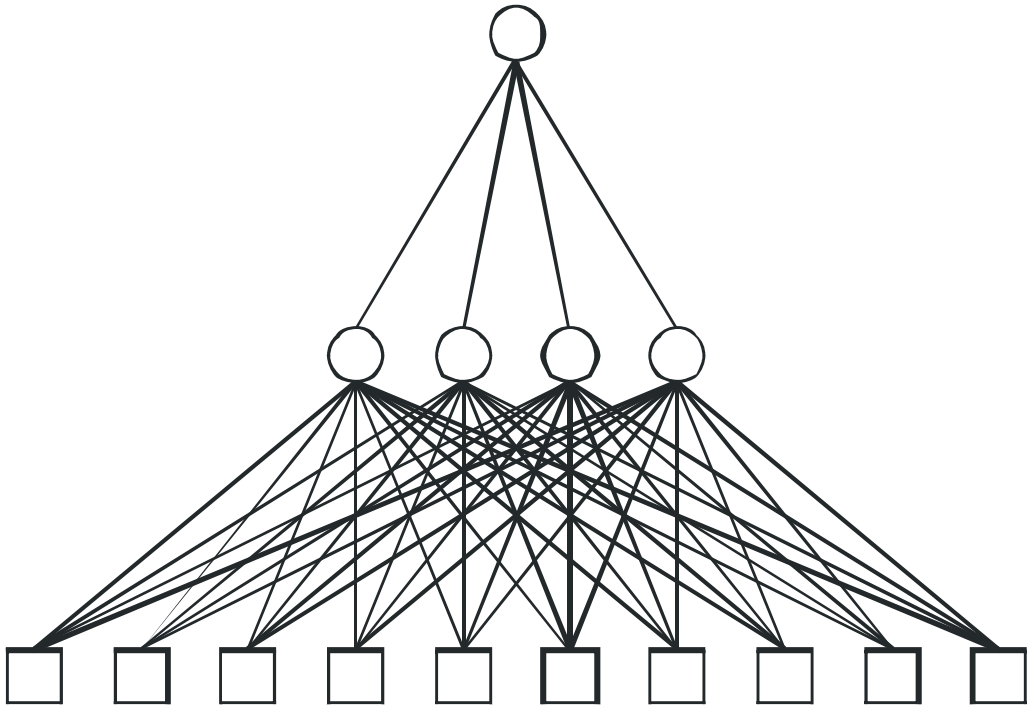
Output units $O_i$

$W_{j,i}$

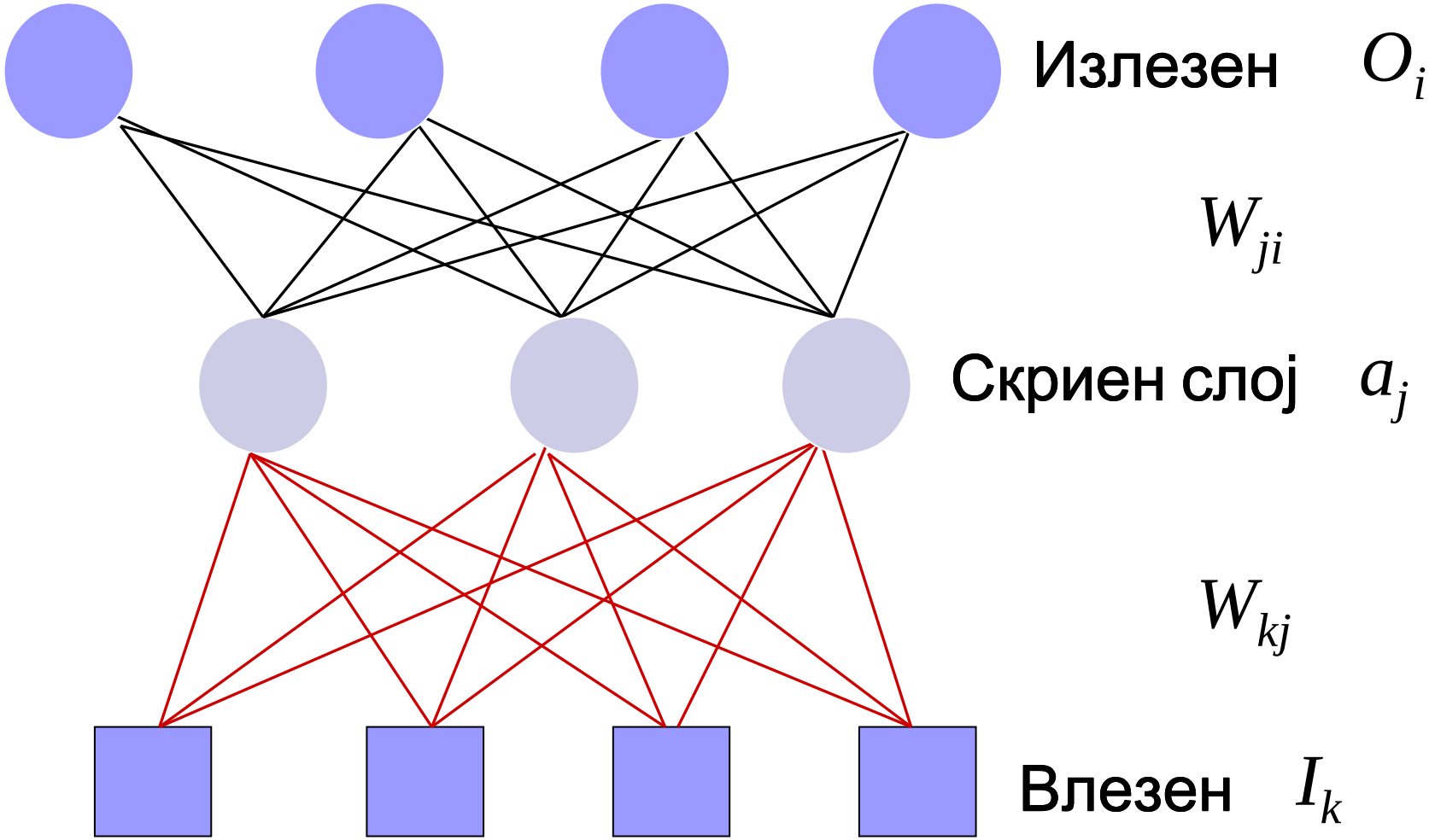
Hidden units $a_j$

$W_{k,j}$

Input units $I_k$



# Back-Propagation



# Грешка кај повеќеслојните НМ

- Да се подели одговорноста за грешката на соодветните тежини

$$Err_i = T_i - O_i$$

$$W_{j,i} \leftarrow W_{j,i} + \alpha \cdot a_j \cdot Err_i \cdot g'(in_i)$$

- по смена: $\Delta_i = Err_i \cdot g'(in_i)$

$$W_{j,i} \leftarrow W_{j,i} + \alpha \cdot a_j \cdot \Delta_i$$

- скриениот јазол $j$ е „одговорен“ за дел од грешката $\Delta_i$ во секој од излезните јазли со кои е поврзан. Грешката $\Delta_i$ се дели според јачината на врската помеѓу скриениот и излезниот јазел и се пропагираат наназад како грешки за невроните од скриеното ниво.

# Распространување (пропагација) на грешката

Правило за распространување

$$\Delta_j = g'(in_j)\sum_i W_{j,i}\Delta_i$$

Промена на тежините
Влез - скриен слој

$$W_{k,j} \leftarrow W_{k,j} + \alpha \cdot I_k \cdot \Delta_j$$

Алгоритам:

- Пресметај ја Δ вредноста за излезните неврони според забележаната грешка
- Почнувајќи од излезното ниво повторувај за секое ниво на мрежата се до влезното ниво:
  - Пропагирај ги Δ вредностите назад низ претходното ниво
  - Измени ги тежините помеѓу двете нивоа

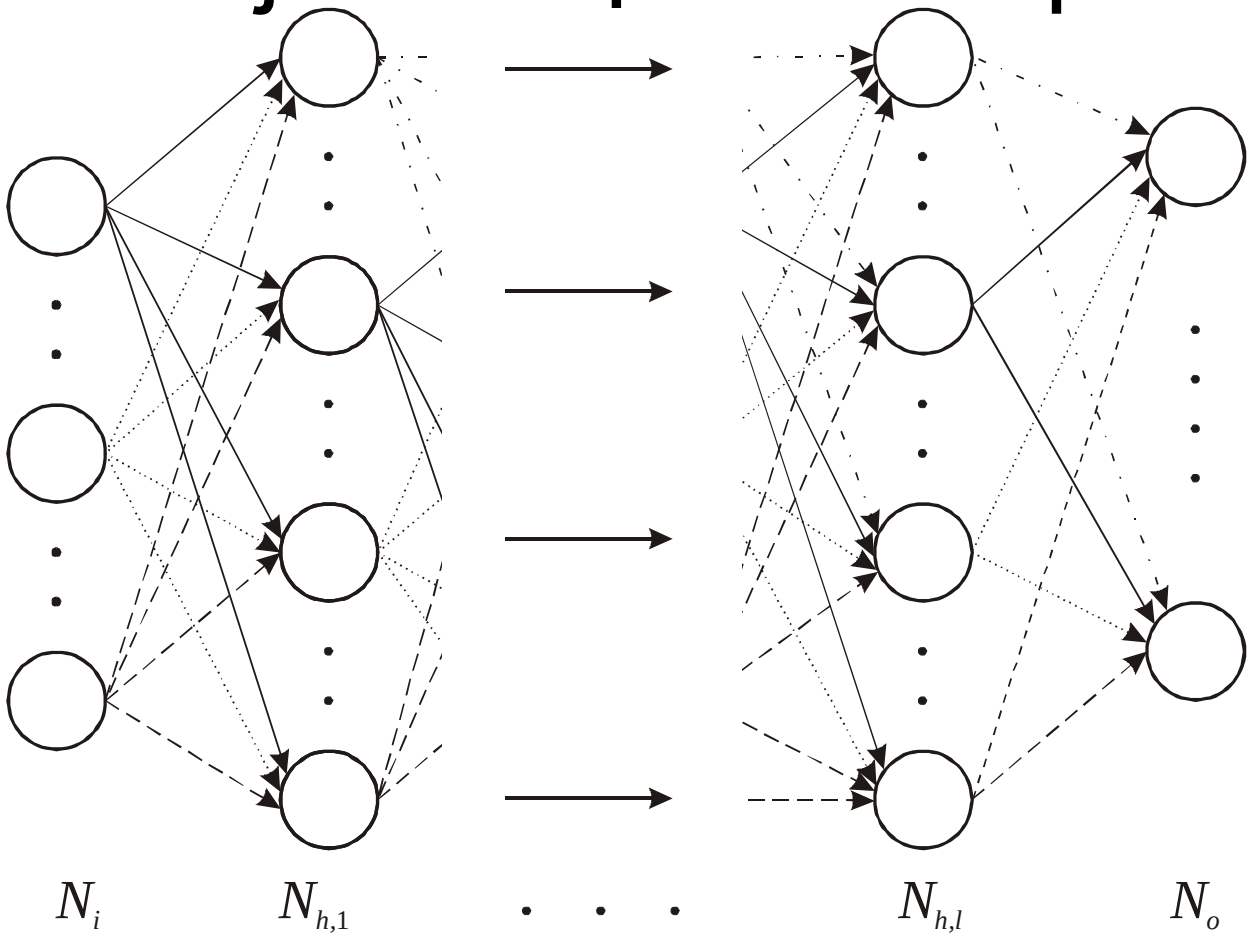
# Back-propagation algorithm*

**function** BACK-PROP-UPDATE(*network*, *examples*, *a)* **returns** a network with modified weights

**inputs:** *network*, a multilayer network
*examples*, a set of input/output pairs
α, the learning rate

**repeat**
**for each** *e* **in** *examples* **do**
/* *Compute the output for this example* */
**O** ← RUN-NETWORK(network, $\mathbf{I}^e$)
/* *Compute the error and A for units in the output layer* */
$\mathbf{Err}^e \leftarrow \mathbf{T}^e - \mathbf{O}$
/* *Update the weights leading to the output layer* */
$W_{j,i} \leftarrow W_{j,i} + \alpha \cdot a_j \cdot Err_i^e \cdot g'(in_i)$
**for each** subsequent layer **in** *network* **do**
/* *Compute the error at each node* */
$\Delta_j \leftarrow g'(in_j)\Sigma_i W_{j,i}\Delta_i$
/* *Update the weights leading into the layer* */
$W_{k,j} \leftarrow W_{k,j} + a \cdot I_k \cdot \Delta_i$
**end**
**end**
**until** *network* has converged
**return** *network*

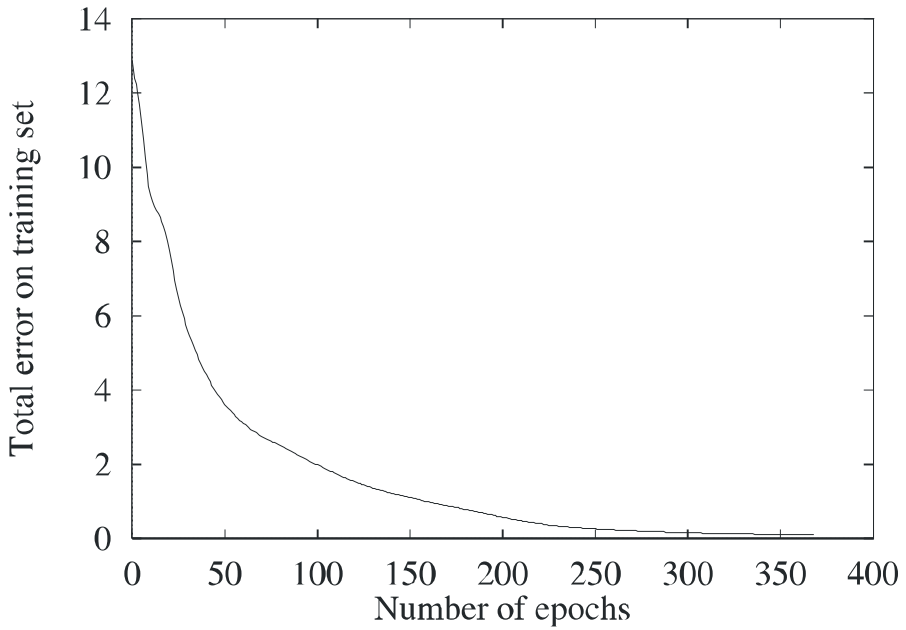
# Повеќеслојни невронски мрежи

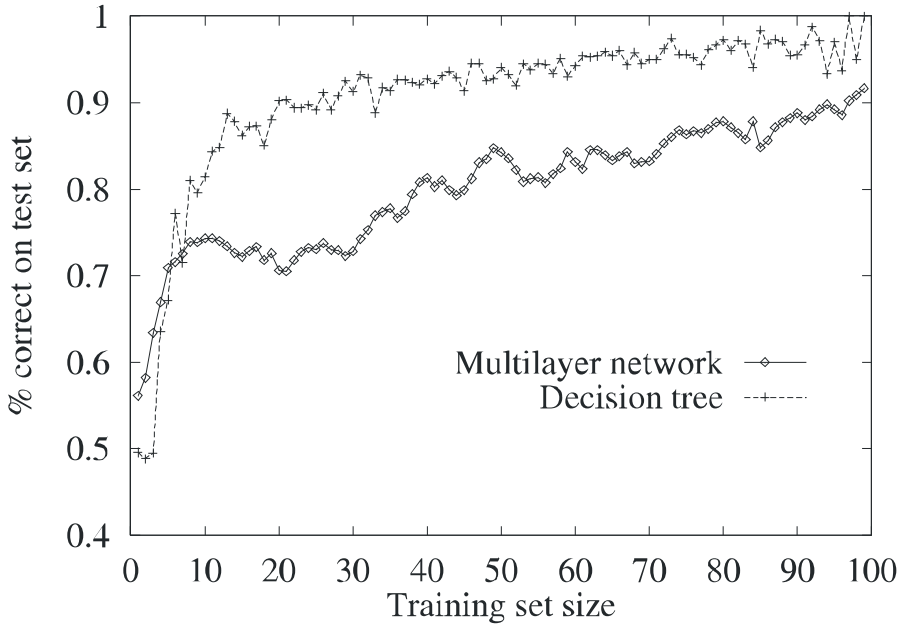



Се покажало дека нема потреба да се оди со повеќе слоеви,
туку истата пресметковна моќ и мемориски капацитет може да се постигне
со зголемување на бројот на јазли во скриениот слој

# Грешка при обуката

- Невронска мрежа со две нивоа (10-4-1) за проблемот „ресторан“



# Карактеристики

- Експресивност
- Пресметковна ефикасност
- Генерализација
- Осетливост на шум (но има и робустни варијанти – отпорни на шум)
- Транспарентност? (за жал – не)

Примени...

# За што се добри невронските мрежи?

- Добри препознавачи на урнеци и робустни класификатори
- Добри при решавање на проблеми кои се премногу сложени за конвенционалните технологии
- Идеални решенија за мноштво проблеми како и за предвидувања

# Прашања?