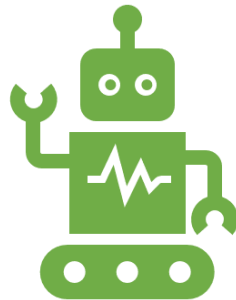




Универзитет „Св. Кирил и Методиј“ во Скопје  
**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И  
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

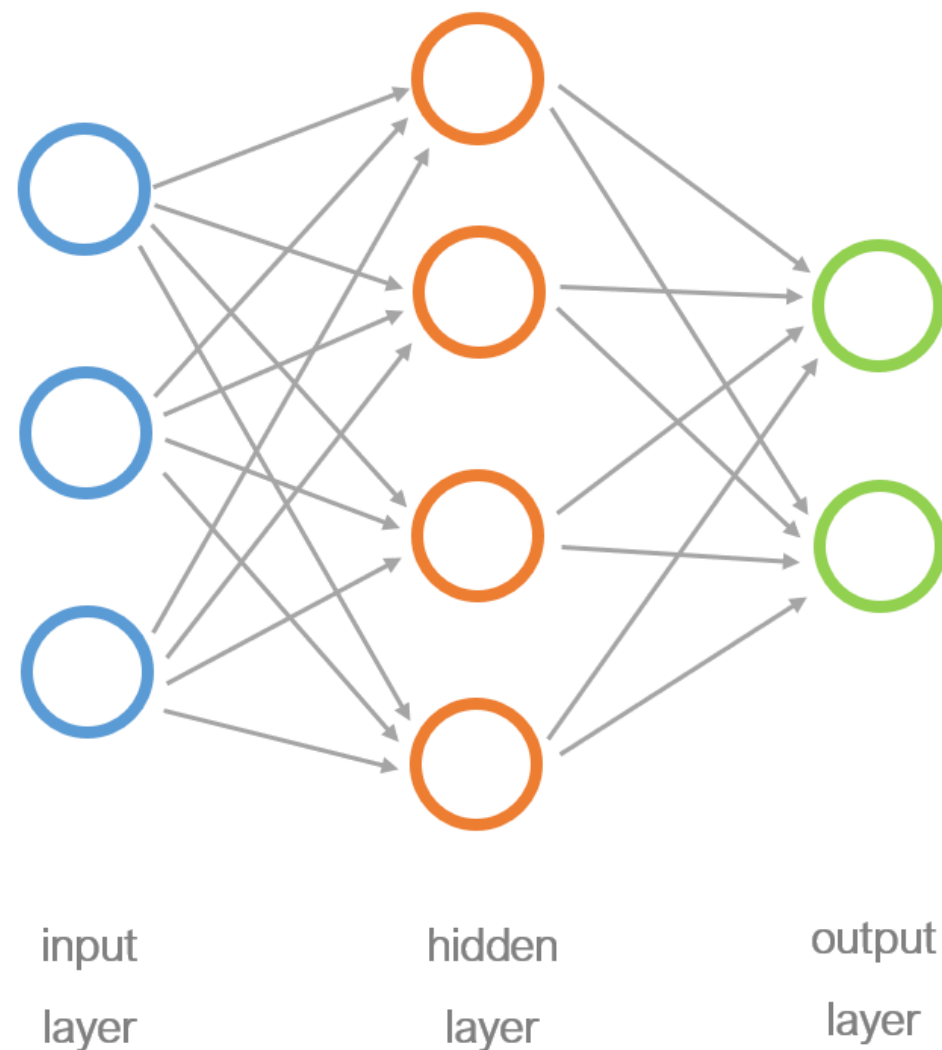
# Невронски мрежи

м-р М артина Тош евска  
м-р Јана Кузманова



**Аудиториски вежби по курсот  
Вештачка интелигенција  
2022/2023**

# Вештачка невронска мрежа – Повеќеслоен перцептрон





# Чекори во алгоритмот

1. Иницијализација на мрежата
2. Тренирање на мрежата
  1. Пропагирање нанапред
  2. Пропагирање на грешката наназад
3. Предвидување



# Иницијализација на мрежата

- Секој неврон има свое множество на тежини кои треба да ги обработува. Постои тежина за секоја влезна врска и една дополнителна за bias.
- Практично е да се иницијализира мрежата со тежини на мали рандом броеви (во опсег 0 до 1).



# Пропагирање нанапред

- Може да се пресмета излезот од невронската мрежа со пропагирање на влезниот сигнал преку секој слој.
- Ова се нарекува пропагирање нанапред.
- Ова е техника која е потребна за генерирање предикции при тренирање.
- Пропагирањето нанапред е составено од три делови:
  - Пресметка на неврон
  - Активација на неврон
  - Пропагација



# Пресметка на неврон

- Првиот чекор е да се пресмета активацијата на еден неврон ако е даден влез.
- Влезот може да биде ред од нашето тренинг множество, како во случајот со скриениот слој. Исто така, може да биде излез од секој неврон од скриениот слој, како што е во излезниот слој.
- Активацијата на невронот се пресметува како тежинска сума на влезовите. Слично како во линеарна регресија.

- $$\textit{activation} = \sum_{i=0}^N (\textit{weight}_i * \textit{input}_i) + \textit{bias}$$

каде ***weight*** е мрежна тежина, ***input*** е влез, ***i*** е индексот на тежина или влез и ***bias*** е специјална тежина која нема влез по кој ќе се помножи (или може да се смета како влез кој е секогаш).



# Активациска функција на невронот

- Кога ќе се активира неврон, треба да се направи трансфер на активацијата за да се добие излезот.
- Ако не се примени активациска функција на невроните, целата невронска мрежа ќе биде редуцирана на група од линеарни функции на влезот на мрежата – по една линеарна функција за секој излезен неврон.
- Затоа, без активациски функции, невронската мрежа нема да може да научи нелинеарни зависности.
- Секој неврон може да се претстави како препознавач на некоја карактеристика (feature), со активација од 0 која означува дека има отсуство на дадената карактеристика.



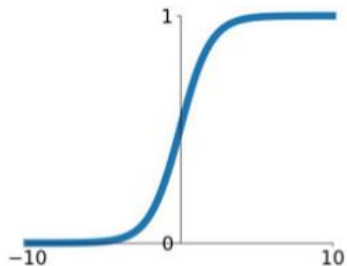
# Активациска функција на невронот (2)

- Може да се користат различни активациски функции за кои има дефинирано извод. Традиционално се користи **softmax активациската функција**, но исто така може да се користи  $\tanh$  (hyperbolic tangent) функцијата. Во последните години, се користи и Rectified Linear Unit (ReLU) активациска функција во големи мрежи со длабоко учење.
- Сигмоид функцијата е во облик на буквата S, и се нарекува и логистичка функција. Може да прими било каков влез и резултира во број помеѓу 0 и 1 на S-крива. Исто така претставува функција од која лесно може да се пресмета изводот кој ни е потребен за пропагирање на грешката наназад.

# Активациска функција на невронот (3)

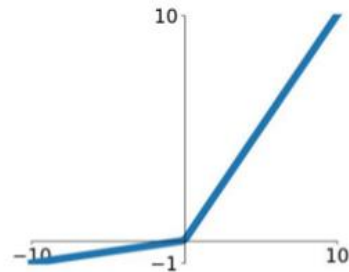
**Sigmoid**

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



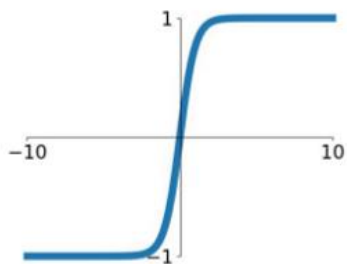
**Leaky ReLU**

$$\max(0.1x, x)$$



**tanh**

$$\tanh(x)$$

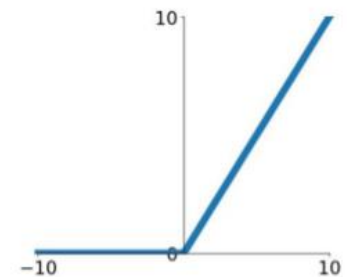


**Maxout**

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

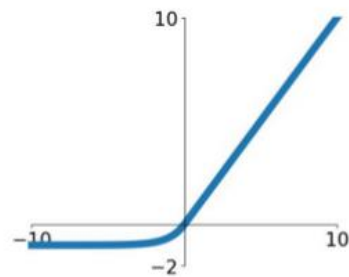
**ReLU**

$$\max(0, x)$$



**ELU**

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





# Алгоритам за пропагирање наназад (Backpropagation)

- Алгоритмот за пропагирање наназад е надгледуван метод за учење за повеќеслојни преносни мрежи во областа на вештачките невронски мрежи.
- Принципот на пристапот со пропагирање наназад моделира дадена функција преку модифицирање на внатрешните тежини на влезните сигнали за да се произведе очекуваниот излезен сигнал. Системот се тренира со користење на надгледуван метод за учење, каде што грешката помеѓу излезот на системот и познатиот очекуван излез се прикажува на системот и се користи за менување на нејзината внатрешна состојба.

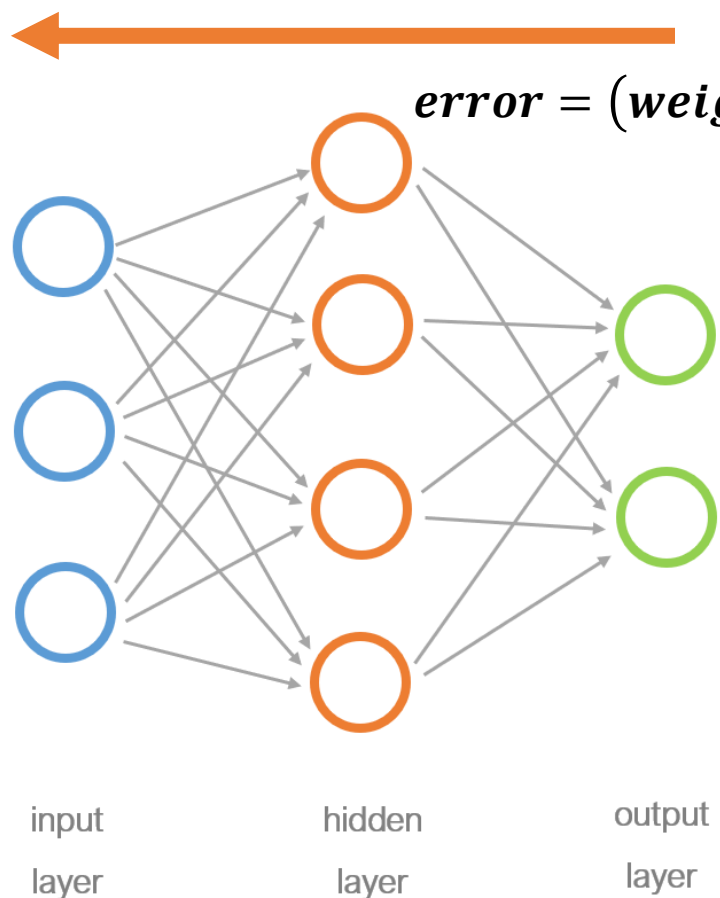


# Пропагирање на грешката наназад

- Алгоритмот со пропагирање наназад го добил своето име според начинот на кој се тренираат тежините.
- Се пресметува грешка помеѓу очекуваните излези и излезите кои се пропагирани нанапред во мрежата со функција на загуба. Оваа грешка се пропагира наназад во мрежата од излезниот слој до скриениот слој, со update на тежините.
- Два дела:
  - Извод на активациската функција
  - Пропагирање на грешката наназад



# Пропагирање на грешката наназад (2)

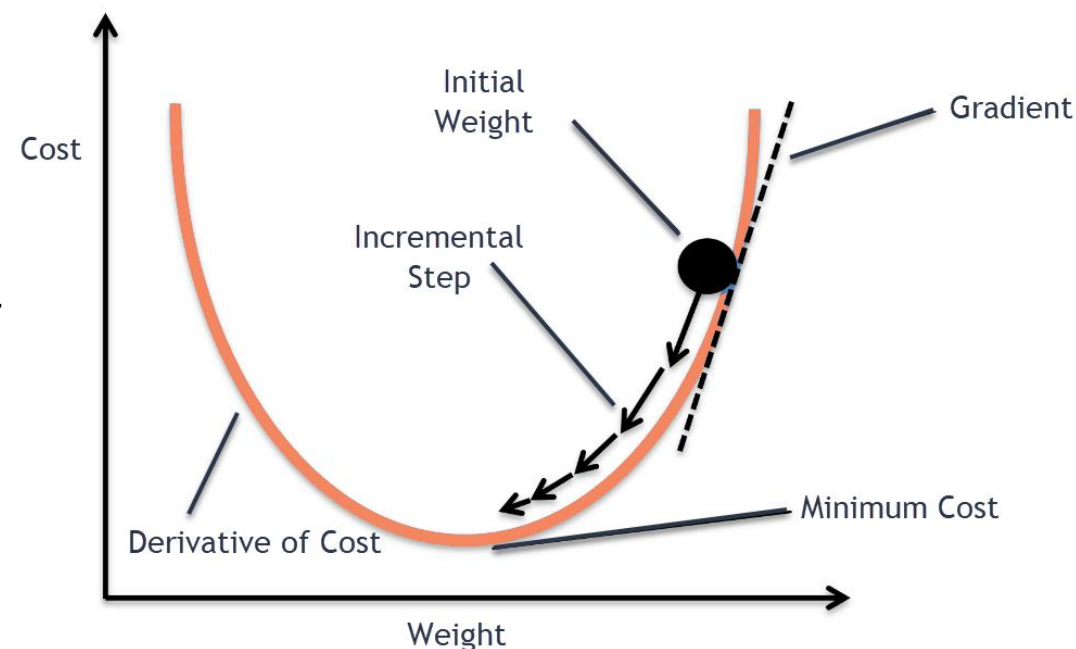


$$error = (weight_k * error_j) * derivative(output)$$

$$error = (expected - output) * derivative(output)$$

# Тренирање на мрежата

- Мрежата се тренира со користење на stochastic gradient descent.
- Ова вклучува повеќе итерации за изложување на тренинг множеството на мрежата и за секој ред од множеството се пропагираат влезовите нанапред, потоа се пропагира грешката наназад, и на крај се прави update на тежините.
- Два дела:
  - Ажурирање на тежини
  - Тренирање на мрежа



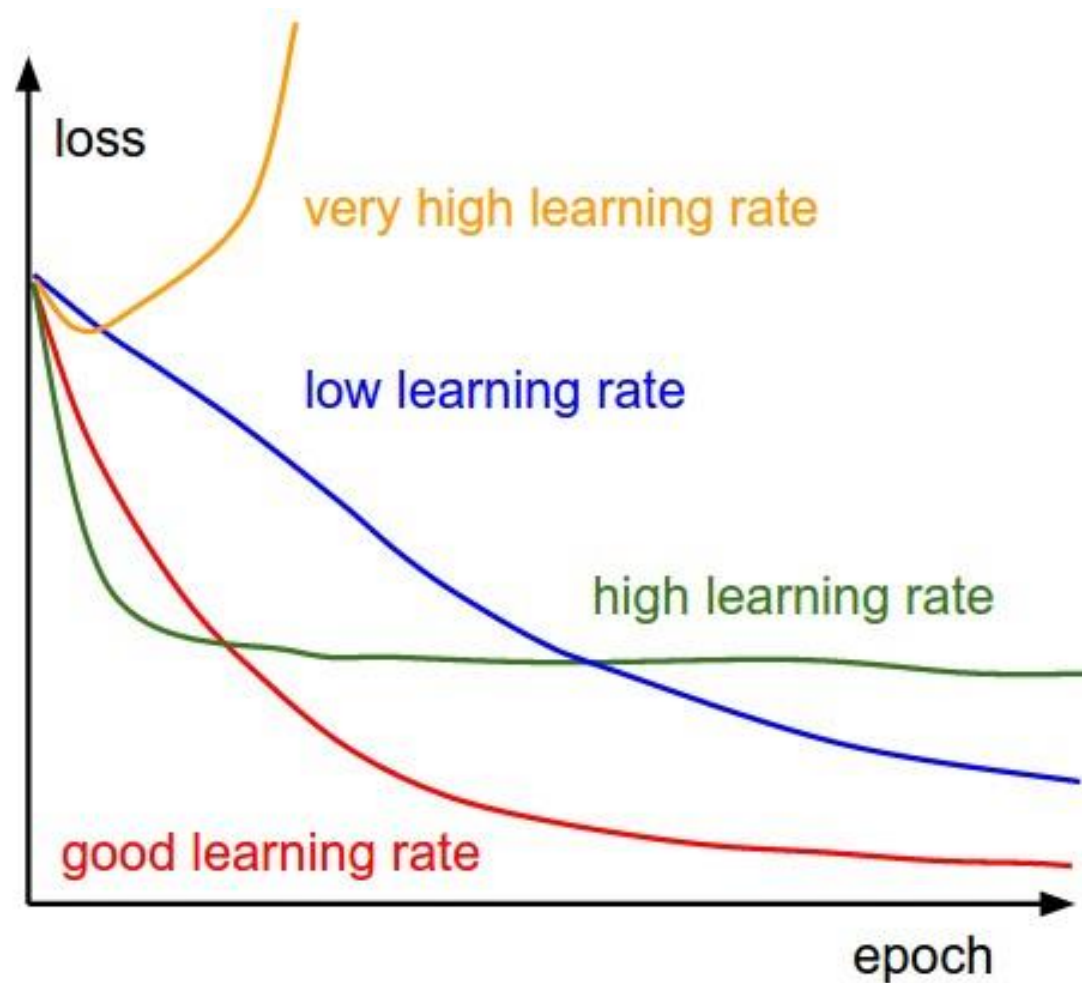


# Ажурирање на тежини

- Кога ќе се пресметаат грешките за секој неврон во мрежата со алгоритмот за пропагирање наназад, тие се користат за ажурирање на тежините.
- Мрежните тежини се ажурираат според:  
$$\textit{weight} = \textit{weight} + \textit{learning\_rate} * \textit{error} * \textit{input}$$
каде ***weight*** е дадена тежина, ***learning\_rate*** е параметар, ***error*** е грешката пресметана со алгоритмот за пропагирање на грешка наназад за невронот и ***input*** е влезната вредност што ја предизвикала грешката.
- Learning rate (ратата на учење) контролира колку да се промени тежината според грешката. На пример, вредност од 0.1 ќе ја ажурира тежината за 10% од вредноста од која може да се ажурира.



# Рата на учење





# Параметри за тренирање

- Број на неврони во скриениот слој
- Активациска функција на невроните
- Рата на учење
- Тип на оптимизација
- Број на епохи за тренирање
- ...



# MLPClassifier

- Метода **fit()** – тренирање на моделот
- Метода **predict()** – прави класификација на запис
- Метода **predict\_proba()** – ги враќа веројатностите запис да припаѓа во секоја класа



## MLPClassifier (2)

- Извлекување на тежините и `bias` вредностите по тренирање на моделот:
  - `coefs_` – листа на матриците со тежини, каде што тежинската матрица на индекс  $i$  ги претставува тежините помеѓу слојот  $i$  и слојот  $i+1$ .
  - `intercepts_` – листа на `bias` вектори, каде што векторот на позиција  $i$  ги претставува `bias` вредностите додадени на слојот  $i+1$ .



# Нормализација на карактеристиките

- **StandardScaler** – стандардизација на карактеристиките со одземање на средната вредност и скалирање на варијансата

$$z = (x - m) / s$$

- **MinMaxScaler** – скалирање на карактеристиките со одреден дефиниран ранг на вредности

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} * (max - min) + min$$

- **MaxAbsScaler** – Скалирање на карактеристиките со нивната максимална апсолутна вредност
- **RobustScaler** – Скалирање на карактеристиките според статистики робусни на аутлаери. Се отстрануваат вредностите на медијаната и се скалираат податоците според интерквartilниот ранг