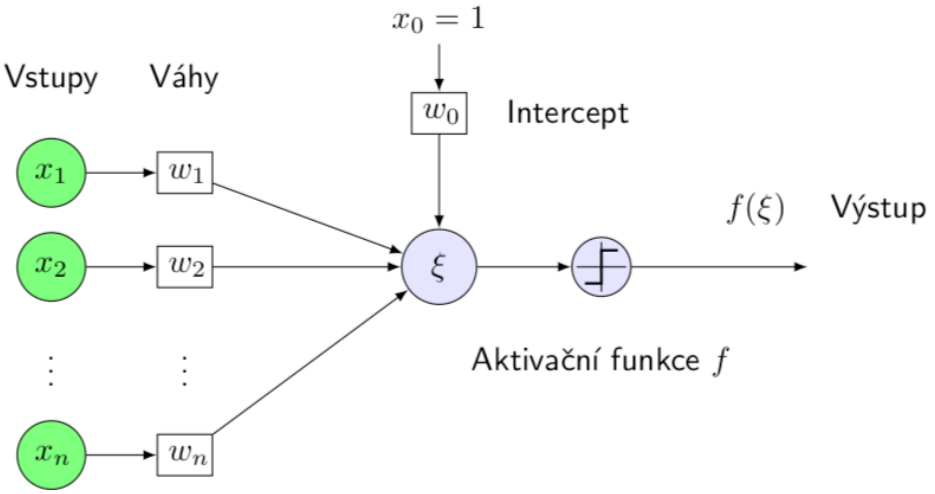
# 22 – Reprezentace znalostí a inference pomocí neuronových sítí

- Biologická inspirace
- klasifikace, regrese, predikce časových řad

## Perceptron

- Nejjednoduší model, který se skládá z 1 umělého neuronu
- Funguje jen pro lineárně separovatelná data (najde pouze lineární fci)
    - Selže například pro XOR
- Výstup neuronu se získá aplikací nelineární **aktivační funkce f** na hodnotu **vnitřního potenciálu** ξ daného součtem vstupů $x_1, \dots, x_n$ pronásobených příslušnýmivahami $w_1, \dots, w_n$ a interceptu $w_0$ (**bias**).




## Vniřní potenciál

- Vstupy $x = (x_1, \dots, x_n)^T$, váhy $w = (w_1, \dots, w_n)^T$, $w_0$ je bias

$$\xi = w_0 + \sum_{i=1}^{n} w_i x_i = \boldsymbol{w}^T \boldsymbol{x} + w_0,$$

## Výstup perceptronu

- Tento výpočet se nazývá **forward pass**

$$\hat{Y} = f(\xi) = f\left(\boldsymbol{w}^T \boldsymbol{x} + w_0\right),$$

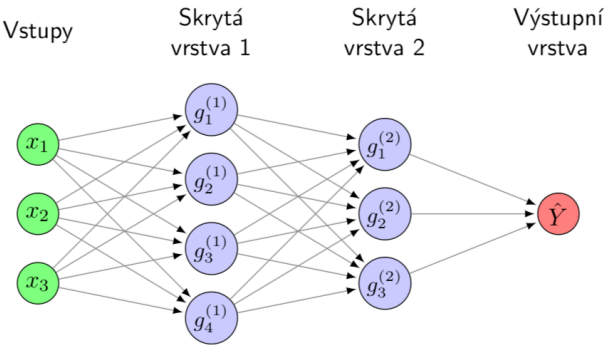
## Inkrementální update vah

- Označuje se jako **backward pass**
- $\hat{Y}$ je predikce, $Y$ je ground truth, μ je learning rate, $x_i$ je jeden sample

$$\begin{aligned} \delta w_i &= \eta(Y - \hat{Y}) x_i, \\ w_i &\leftarrow w_i + \delta w_i, \end{aligned}$$

# Multi Layer Perceptron

- Výstupy neuronů z jedné vrstvy tvoří vstupy neuronů do další vrstvy
- V problému XOR dokáže skrytá vrstva zajistit transformaci do souřadného systému, kde jsou již body separabilní.



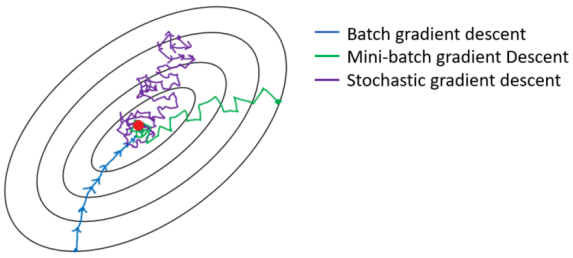

- Třívrstvá NN se dá rozložit jako $g(x) = g^{(3)}\left(g^{(2)}\left(g^{(1)}(x)\right)\right)$

## Trénování MLP

- K učení lze využít i black-box metody (genetické algoritmy)
- Revoluce v učení NN: **back-propagation** (zpětné šíření chyby)
- Vyžaduje, aby byla celá NN diferencovatelná podle parametrů sítě (vah w) podle kterých minimalizujeme loss funkci:

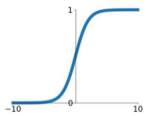
$$J(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - g(\boldsymbol{x}_i)\right)^2$$

- Gradient descend – iterativě upravujeme parametry sítě (váhy) a minimalizujeme tak chybu sítě. Než upravíme váhy, můžeme využít různá množství dat (**batch size**)
  - Batch training (chyba se napočítá z celé trénovací množiny)
  - Minibatches (chyba se napočítá z části trénovací množiny)
  - Online training (chyba se napočítá z 1 samplu)

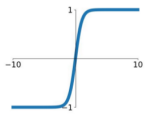



- **Aktivační funkce**
  - cílem je dodat nelinearitu do sítě (bez ní je NN lineární regrese bez ohledu na počet vrstev)
  - pro multiclass klasifikaci SoftMax (predikce psti, že vstup náleží do třídy)

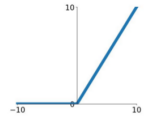
**Sigmoid**
$\sigma(x) = \frac{1}{1+e^{-x}}$



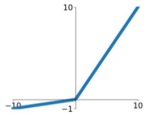
**tanh**
$\tanh(x)$



**ReLU**
$\max(0, x)$

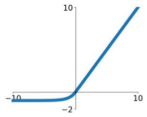


**Leaky ReLU**
$\max(0.1x, x)$



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

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



## Matematický model vícevrstvé neuronové sítě

- Uvažujme $l$ vrstvou neuronovou síť a označme $n_1, \ldots, n_l$ počty neuronů v jednotlivých vrstvách. Dále označně počet vstupních proměnných jako $n_0$.
- Uvažujme $i$tou vrstvu této sítě. Výstup $j$-tého neuronu může reprezentovat funkcí $g_j^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}$, která má na vstupu výstupy $n_{i-1}$ neuronů z předchozí vrstvy.
- Interně se $g_j^{(i)}(\boldsymbol{x})$ opět počítá jako $f(\boldsymbol{w}^T \boldsymbol{x} + w_0)$, kde $f$ je aktivační funkce daného neuronu a $w_0, w_1 \ldots, w_n$ jsou jeho váhy.
- $i$-tou vrstvu této neuronové sítě jako celek pak můžeme chápat jako vícehodnotovou funkci $\boldsymbol{g}^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$, kde $\boldsymbol{g}^{(i)} = \left(g_1^{(i)}, \ldots, g_{n_i}^{(i)}\right)^T$.
- Celá neuronová síť při dopředném chodu je tedy reprezentována funkcí $\boldsymbol{g} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_l}$, která vznikne složením jednotlivých vrstev

$$\boldsymbol{g} = \boldsymbol{g}^{(1)} \circ \boldsymbol{g}^{(2)} \circ \ldots \circ \boldsymbol{g}^{(l-1)} \circ \boldsymbol{g}^{(l)}.$$

- Např. pro $l = 3$ tak máme

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}^{(3)}\left(\boldsymbol{g}^{(2)}\left(\boldsymbol{g}^{(1)}(\boldsymbol{x})\right)\right).$$





## Dávkové učení neuronové sítě

- Máme neuronovou síť s parametry $\boldsymbol{w} = (w_1, \ldots, w_m)^T$ a trénovací data $(Y_1, \boldsymbol{x}_1), \ldots, (Y_N, \boldsymbol{x}_N)$.
- Inicializujeme všechny váhy $\boldsymbol{w}$ jako malá náhodná čísla.
- Opakujeme dokud nejsou splněna kritéria zastavení:
  - Položme $J(\boldsymbol{w}) = 0$.
  - Pro každou trénovací dvojici $(Y_i, \boldsymbol{x}_i)$:
    - Spočteme $\hat{Y}_i$ v bodě $\boldsymbol{x}_i$.
    - Provedeme přepočet celkové chyby,

$$J(\boldsymbol{w}) \leftarrow J(\boldsymbol{w}) + \frac{1}{N}\left(Y_i - \hat{Y}_i\right)^2.$$

  - Spočteme gradient

$$\nabla_{\boldsymbol{w}} J = \left(\frac{\partial J}{\partial w_1}, \ldots \frac{\partial J}{\partial w_m}\right)^T.$$

  - Provedeme přepočet vah

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} J.$$