Neuronové sítě

Tomáš Vlk (vlktoma5@fit.cvut.cz)

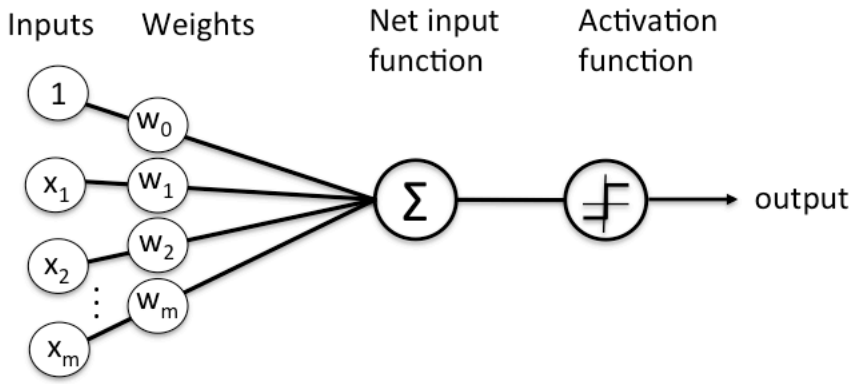
May 28, 2019

Úvod

Inspirováno neuronovými sítěmi v biologických systémech. Neuron tvoří základní stavební jednotku v neuronových sítích. Obvykle přijímá více vstupů a má pouze jeden výstup. První model neuronu již v roce 1943.

Jednovrstvý percepton

Nejjednoduším modelem neuronové klasifikační sítě je jednovstrvý percepton. Výstup se získává aplikací nelineární aktivační funkce f na vnitřní potenciál Σ. Vnitřní potenciál je daný součtem vstupů $x_1, \ldots, x_m$ pronásobených vahami $w_0, \ldots, w_m$ a interceptu w_0.



Schematic of Rosenblatt's perceptron.

Vnitřního potenciál tedy je $\Sigma = w_0 + \sum_{i=1}^{m} w_i x_i = \boldsymbol{w}^T \boldsymbol{x} + w_0$, kde $\boldsymbol{x} = (x_1, \ldots, x_m)^T$ a $\boldsymbol{w} = (w_1, \ldots, w_m)^T$.

Výstup neuronu je $f(\Sigma) = f(\boldsymbol{w}^T \boldsymbol{x} + w_0)$. Aktivační funkce je v případě perceptonu skoková, tedy:

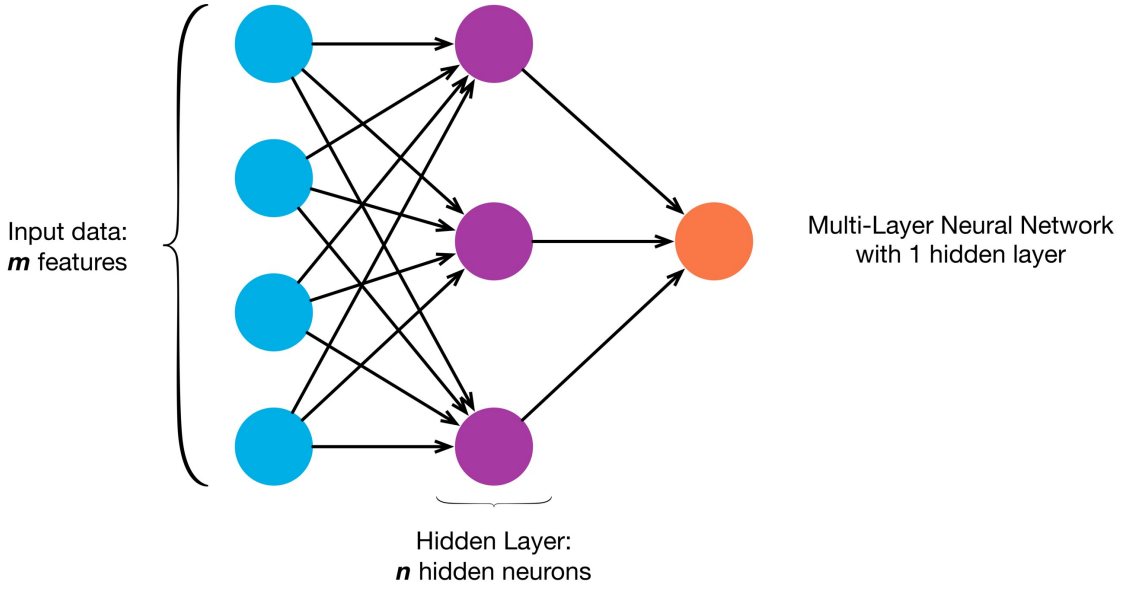
$$f(\Sigma) = \begin{cases} 1 & \text{když } \Sigma \geq 0 \\ 0 & \text{když } \Sigma < 0 \end{cases}$$

Aktivace neuronu tedy proběhne, když $\sum_{i=1}^{m} w_i x_i \geq -w_0$. Proto je občas hodnota w_0 nazývána prahová.

Výpočet výstupu neuronu se nazývá **dopředný chod**[1]. Výpočet vah $w_1, \ldots, w_m$ se nazývá **zpětný chod**[2] a probíhá následovně.

Na základě dopředného chodu spočítáme chybu predikce pomocí rovnice $\delta w_i = \eta(Y - \hat{Y})x_i$, kde $\hat{Y}$ je hodnota predikce v bodě $\mathbf{x}$ a Y je skutečná hodnota. Následně provedeme update vah jako $w_i \leftarrow w_i + \delta w_i$.

Vícevrstvá síť



Základním rozšířením jednoho neuronu je vícevrstvý percepton[3]. Skládá se z více vrstev, výstupy jedné vrstvy jsou vstupy do další vrstvy. Krom vstupní a výstupní vrstvy se všechny vrstvy nazívají skryté.

Uvažujme l vsrtvou neuronovou síť a označme $n_1, \ldots, n_l$ počty neuronů v jednotlivých vstvách. Počet vstupních proměných je značen jako n_0. Poté výstup j-tého neuronu z i-té vrstvy, lze napsat jako $g_j^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}$.

Pro i-tou vrstvu bude tedy výstup roven funkci $g^{(i)} : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$, kde $\boldsymbol{g}^{(i)} = (g_1^{(i)}, \ldots, g_{n_i}^{(i)})^T$. A tedy celá neuronová síť v dopředném chodu je rovna funkci $\boldsymbol{g} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_l}$, jenž vznikne složením jednotlivých vrstev $\boldsymbol{g} = \boldsymbol{g^{(1)}} \circ \boldsymbol{g^{(2)}} \circ \cdots \circ \boldsymbol{g^{(l-1)}} \circ \boldsymbol{g^{(l)}}$.

Obecně se u neuronových sítí preferuje větší počet vrstev oproti většímu počtu neuronů v jedné vrstvě. Učení vícevrstvých neuronových sítí probíhá pomocí **zpětného šíření**[4]. Pro tento přístup je požadována diferenco-

[1] Angl. forward pass
[2] Angl. backward pass
[3] Angl. multilayer percepton, abr. MLP
[4] Angl. back-propagation

vatelnost aktivační funkce.

Aktivační funkce

Příklady pár diferencovatelných aktivačních funkcí.

- **Logistická funkce, sigmoida**
$$f(\Sigma) = \frac{1}{1 + e^{-\Sigma}}$$
- **Hyperbolický tangens**
$$f(\Sigma) = tanh(\Sigma) = \frac{e^{\Sigma} - e^{-\Sigma}}{e^{\Sigma} + e^{-\Sigma}}$$
- **Oříznutá linearní funkce - RELU**
$$f(\Sigma) = max(0, \Sigma) = \begin{cases} x & \text{pro } \Sigma \geq 0, \\ 0 & \text{pro } \Sigma < 0. \end{cases}$$

Učení přes gradient descend

Provádí se minimalizace chyby měřené pomocí kvadratické ztrátové funkce. Minimalizuje se tedy

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

vzhledem k parametrům sítě $\boldsymbol{w}$.

Existuje více přístupů jak často prováděd aktualizaci. Jsou to například:

- **Batch training** - chyba se počítá přes celou trénovací množinu
- **Mini batch training** - chyba se počítá přes několik bodů
- **Online training** - chyba se počítá po každém bodu

Nyní ukážeme jak se při gradientním sestupu postupuje. Přepokládejme že máme neuronovou síť s parametry $\boldsymbol{w} = (w_1, \ldots, w_m)^T$ a trénovací data $(\boldsymbol{Y}_1, \boldsymbol{x}_1), \ldots, (\boldsymbol{Y}_N, \boldsymbol{x}_N)$. Začneme incializací vah na náhodná malá čísla. Poté opakujeme následující proces.

Položíme $J(\boldsymbol{w}) = 0$. Poté pro každou trénovací dvojici $(\boldsymbol{Y}_i, \boldsymbol{x}_i)$, spočteme $\hat{Y}_i$ v bodě x_i a provedeme přepočet celkové chyby jako $J(\boldsymbol{w}) \rightarrow J(\boldsymbol{w}) + \frac{1}{N}(Y_i - \hat{Y}_i)^2$. Tento krok se dělá pouze kvůli zjištění jak velká je chyba, sám o sobě není nutný pro výpočet gradientu. Následně spočteme gradient jako, $\nabla_w J = (\frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_m})^T$. Nakonec provedeme update vah, pomocí $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\nabla_w J$.