

Neuronové sítě

Tomáš Vlček (vlktoma5@fit.cvut.cz)

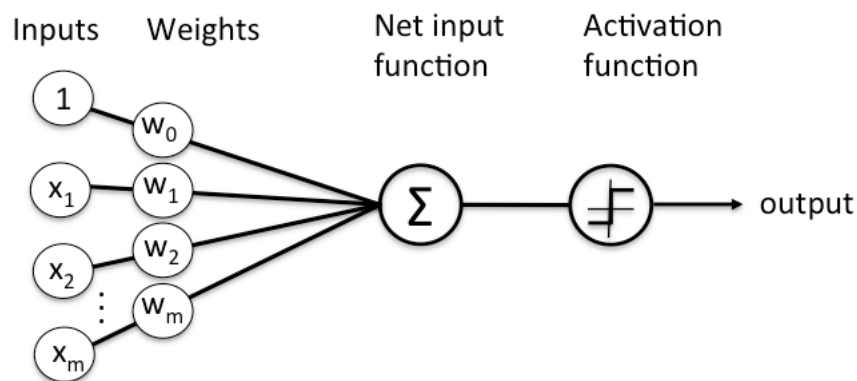
May 29, 2019

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 perceptron

Nejjednodušším modelem neuronové klasifikační sítě je jednovrstvý perceptron. Výstup se získává aplikací nelineární aktivační funkce f na vnitřní potenciál Σ . Vnitřní potenciál je daný součtem vstupů x_1, \dots, x_m pronásobených vahami w_0, \dots, w_m a interceptu w_0 .



Schematic of Rosenblatt's perceptron.

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

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

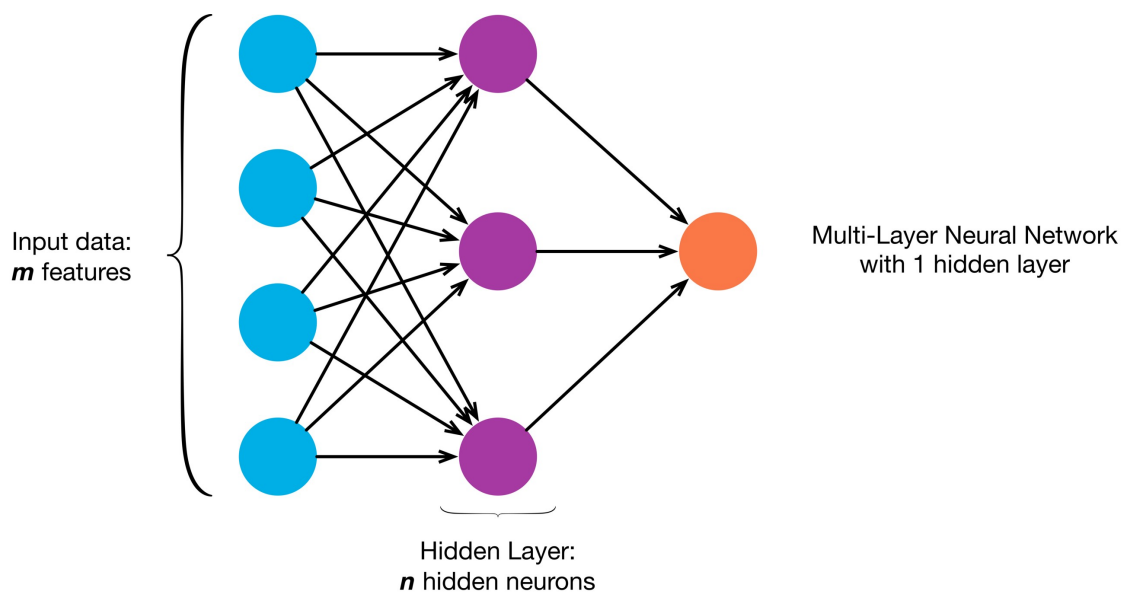
$$f(\Sigma) = \begin{cases} 1 & \text{když } \Sigma \geq 0 \\ 0 & \text{když } \Sigma < 0 \end{cases}$$

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, \dots, 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 perceptron³. Skládá se z více vrstev, výstupy jedné vrstvy jsou vstupy do další vrstvy. Krom vstupní a výstupní vrstvy se všechny vrstvy nazývají skryté.

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

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

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

¹ Angl. forward pass

² Angl. backward pass

³ Angl. multilayer perceptron, 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 lineární 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

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

vzhledem k parametrům sítě \mathbf{w} .

Existuje více přístupů jak často provádět 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. Předpokládejme že máme neuronovou síť s parametry $\mathbf{w} = (w_1, \dots, w_m)^T$ a trénovací data $(\mathbf{Y}_1, \mathbf{x}_1), \dots, (\mathbf{Y}_N, \mathbf{x}_N)$. Začneme inicializací vah na náhodná malá čísla. Poté opakujeme následující proces.

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