



Hluboké učení v teorii i praxi

Michaela Walterová

Jednoocí slepým, 10. a 17.10.2018

Hluboké učení v kontextu

Reprezentační učení: automatické definování **příznaků** (features),
na základě kterých se hledá vztah mezi vstupem a výstupem

Učení s učitelem (Supervised learning):

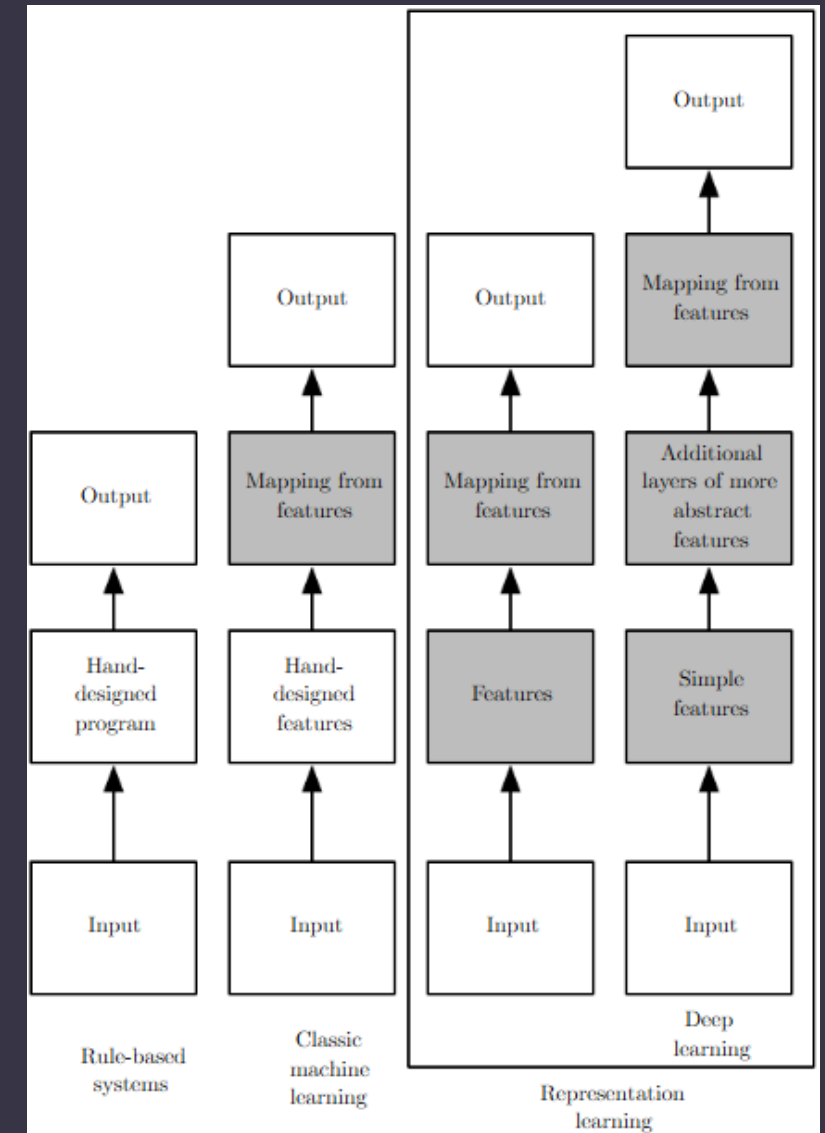
Známe správné odpovědi, minimalizujeme ztrátovou funkci

Učení bez učitele (Unsupervised learning):

Neznáme správné odpovědi, algoritmus sám dělí data podle
příznaků

Zpětnovazební učení (Reinforcement learning):

Hledání nejlepší strategie, interakce agenta a prostředí



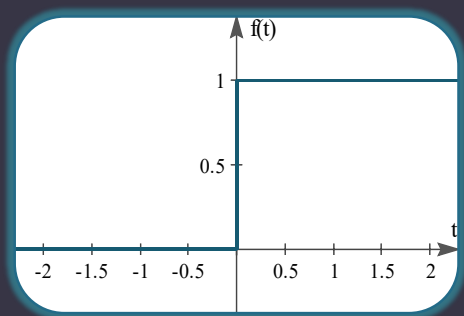
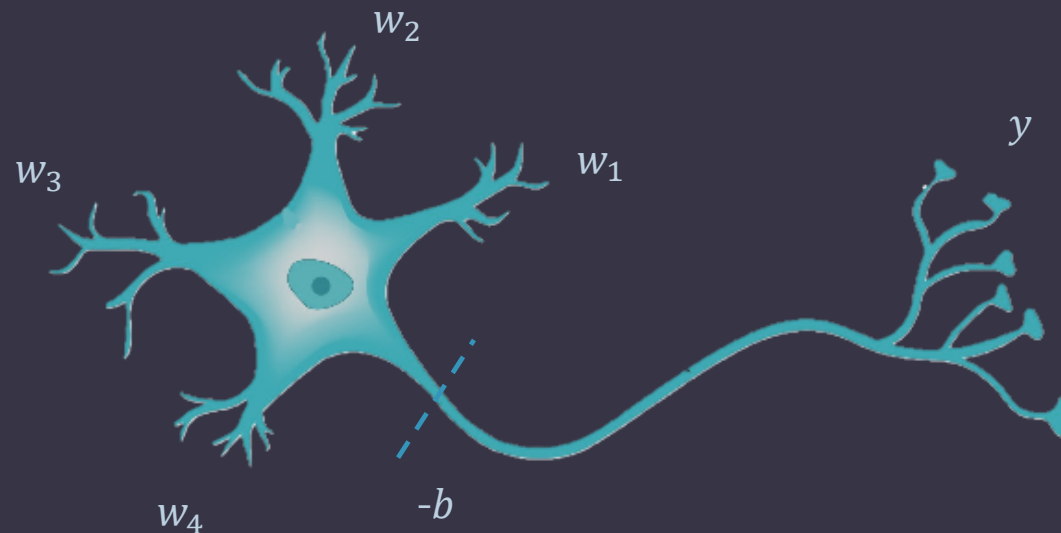
Základní buňky

Neuron = skalární funkce

$$y = h(\mathbf{w}^T \mathbf{x} + b)$$

Parametry modelu jsou **váhy** w_i a **prahy** b každého neuronu

Za **aktivační funkci** h můžeme dosadit například ...

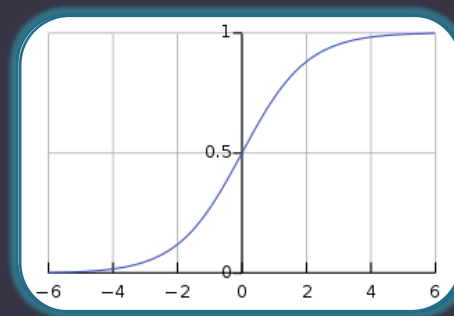
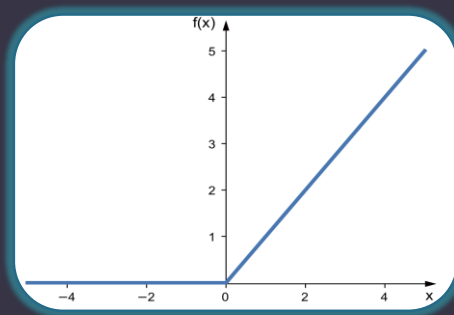


Heavisideovu funkci

$$h(z) = \begin{cases} 1 & \text{pro } z > 0 \\ 0 & \text{pro } z \leq 0 \end{cases}$$

ReLU

$$h(z) = \max(0, z)$$

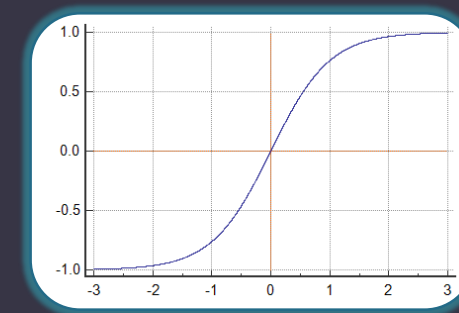


Logistickou funkci
(sigmoid)

$$h(z) = \frac{1}{1 + e^{-z}}$$

Hyperbolický tangens

$$h(z) = \tanh(z)$$

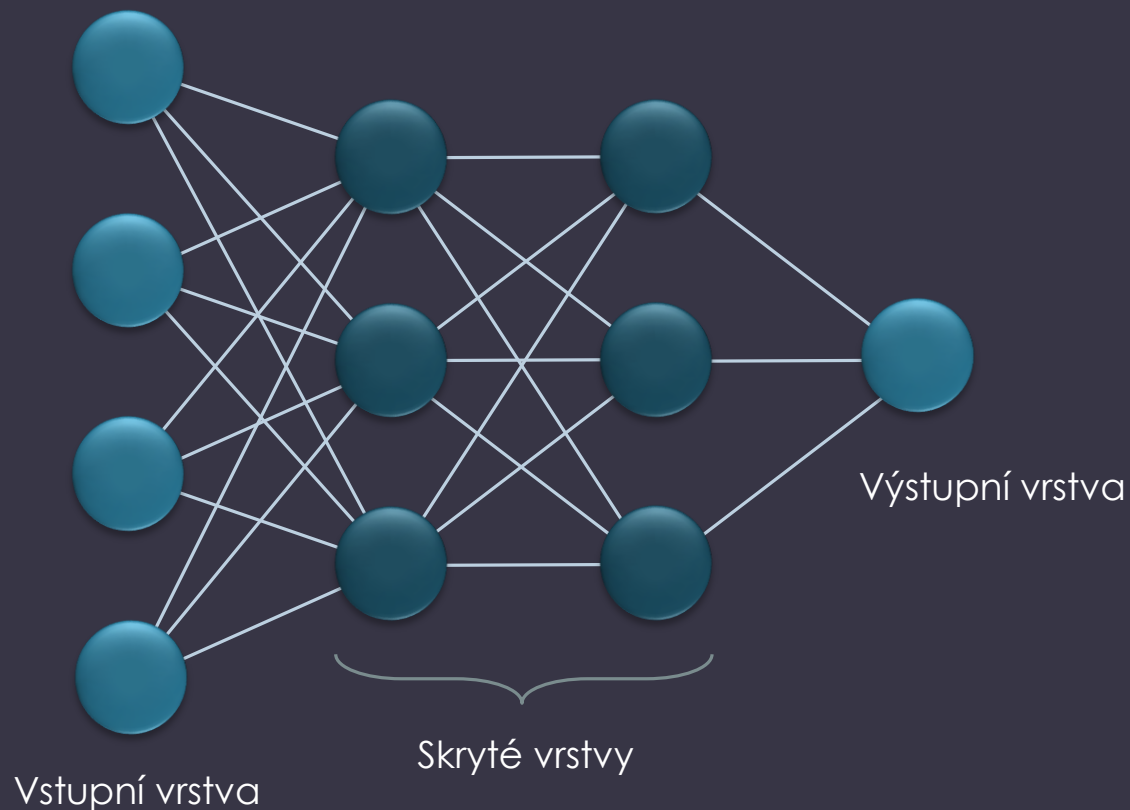


Neuronová síť

Nelineární aproximační funkce

$$\mathbf{y} = f^{(1)}(f^{(2)}(\dots f^{(n)}(\mathbf{x})))$$

popsaná sadou hyperparametrů a parametry w_{ij} , b_i



Výstup z předchozí vrstvy je vstupem pro následující vrstvu

Hyperparametry: počet vrstev, šířka vrstev, aktivační funkce, inicializace vah a prahů, rychlost učení, metody regularizace ...

V dopředných sítích obvykle stačí jediná široká skrytá vrstva (viz univerzální aproximační teorém; Cybenko, 1989)

Učení sítě: algoritmus zpětné propagace

Minimalizace ztrátové funkce $\mathcal{L}(\mathbf{y}(\mathbf{x}; \mathbf{W}, \mathbf{b}), \mathbf{y}^*)$ gradientovou metodou v prostoru parametrů sítě $\theta \leftarrow \theta - \lambda \frac{1}{n} \sum_{i=1}^n \frac{\partial \mathcal{L}^i}{\partial \theta}$

Hyperparametr: rychlost učení λ

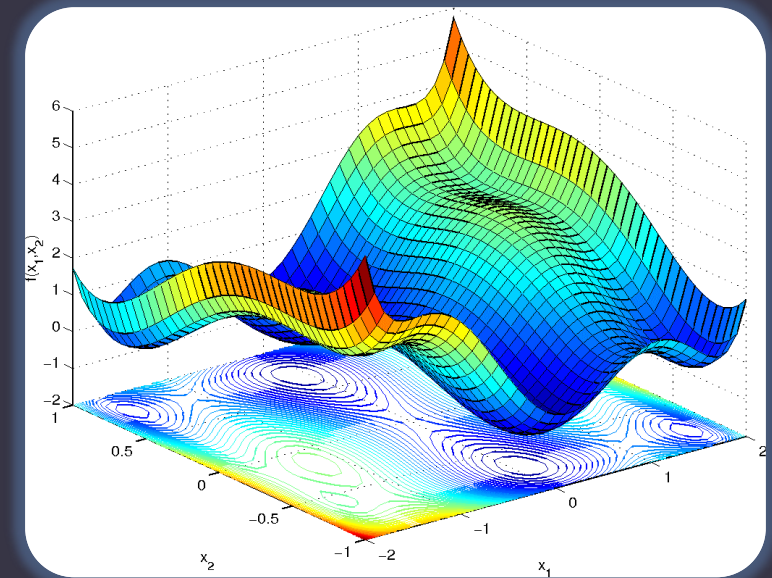
Potřeba velkého množství trénovacích dat + vyhodnocování přesnosti na jiných, neznámých datech

Příliš malá síť nemusí umět postihnout všechny rysy dat

Příliš velká síť se může příliš přizpůsobit trénovacím datům → přetrénování

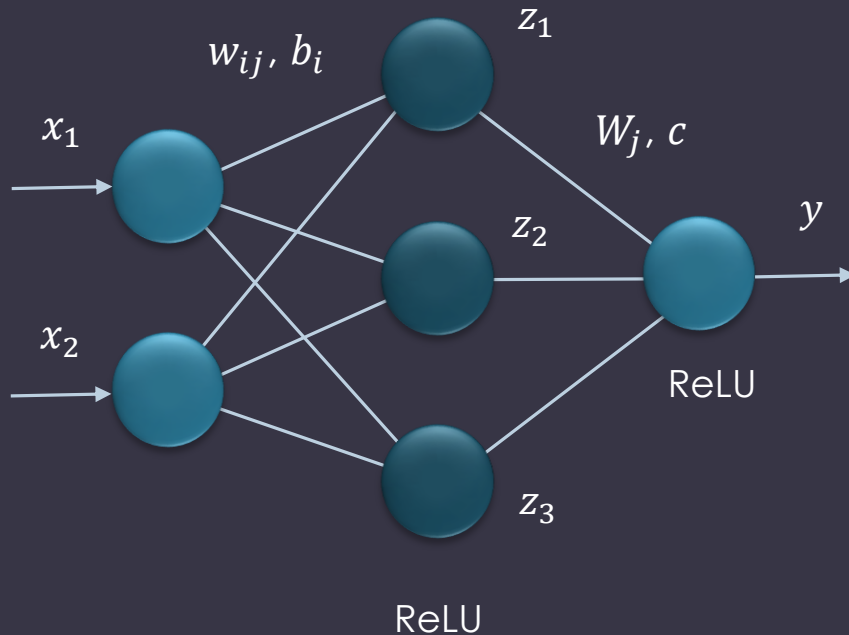
Regularizace: víc dat, „zapomínání“ (dropout), L_2 regularizace...

menší síť, včasné zastavení



Ilustrační příklad: regrese

Trénovacími daty jsou dvojice souřadnic (x_1, x_2) a výsledná hodnota y^* ; síť je popsána 13 parametry (9 vah + 4 prahy)



1) Dopředný chod: vyhodnocení všech neuronů

$$\check{z}_i = \sum_{j=1}^2 w_{ij}x_j + b_i, \quad i = 1, 2, 3 \quad z_i = \max(0, \check{z}_i)$$

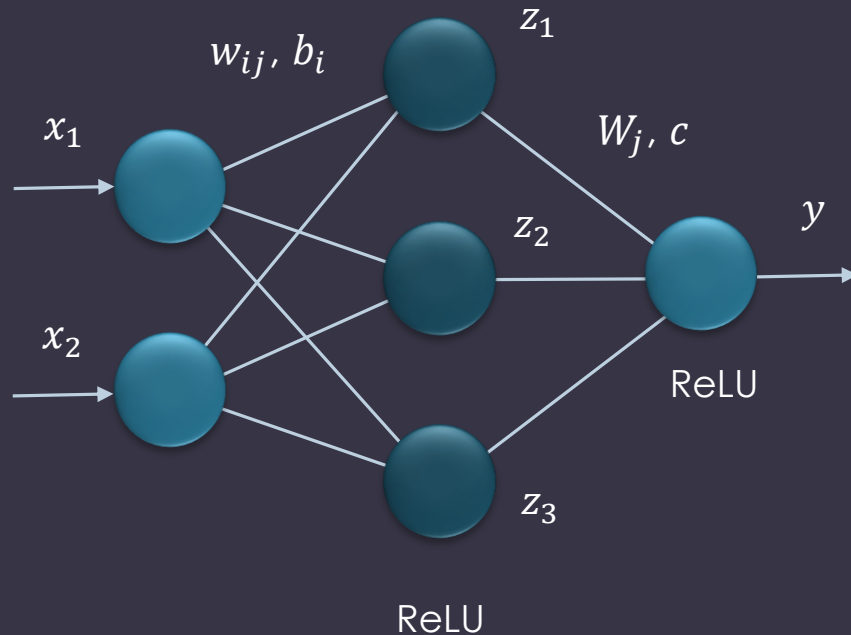
$$\check{y} = \sum_{j=1}^2 W_j z_j + c \quad y = \max(0, \check{y})$$

2) Výpočet chyby/ztrátové funkce

$$\mathcal{L}(y(\mathbf{w}, \mathbf{W}, \mathbf{b}, c), y^*) = (y - y^*)^2$$

Ilustrační příklad: regrese

Trénovacími daty jsou dvojice souřadnic (x_1, x_2) a výsledná hodnota y^* ; síť je popsána 13 parametry (9 vah + 4 prahy)



3) Zpětná propagace: výpočet parciálních derivací

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} \quad \frac{\partial \mathcal{L}}{\partial y} = 2(y - y^*) \quad \frac{\partial y}{\partial \hat{y}} = \begin{cases} 1 & \text{pro } \hat{y} > 0 \\ 0 & \text{pro } \hat{y} < 0 \end{cases}$$

└─ derivace výstupní (aktivační) funkce

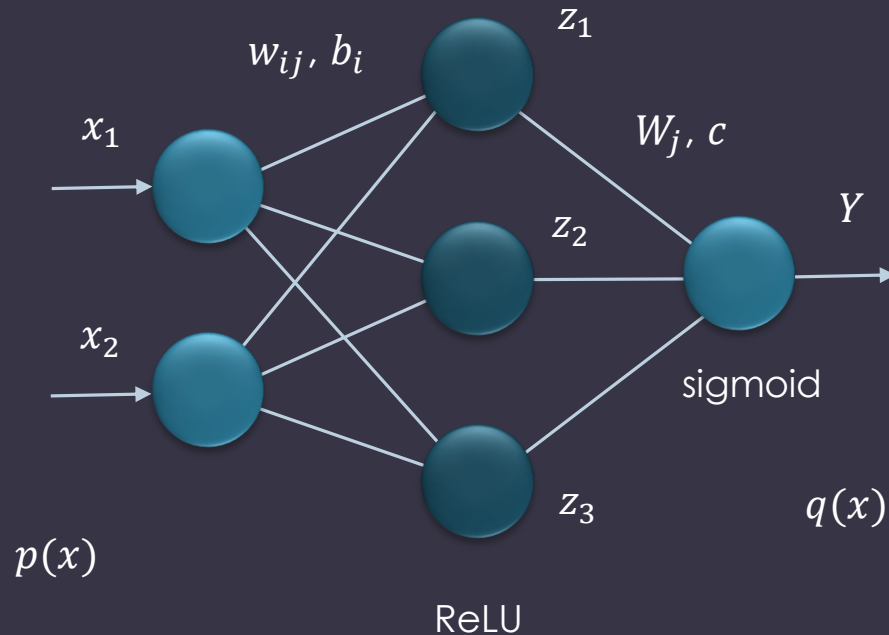
$$\frac{\partial \hat{y}}{\partial c} = 1 \quad \frac{\partial \hat{y}}{\partial W_i} = z_i \quad \frac{\partial \hat{y}}{\partial b_i} = W_i \frac{\partial z_i}{\partial \hat{z}_i} \quad \frac{\partial \hat{y}}{\partial w_{ij}} = W_i \frac{\partial z_i}{\partial \hat{z}_i} x_j$$

4) Úprava parametrů

$$\theta \leftarrow \theta - \lambda \frac{\partial \mathcal{L}}{\partial \theta}, \quad \text{kde} \quad \theta = c, W_j, b_i, w_{ij}$$

Ilustrační příklad: klasifikace

Trénovacími daty jsou dvojice souřadnic (x_1, x_2) a kategorie Y^* ; síť je popsána 13 parametry (9 vah + 4 prahy)



Uvažujme pouze dvě kategorie $\{0,1\}$ – výstupní neuron popisuje pravděpodobnost kategorie „1“

$$\hat{y} = \sum_{j=1}^2 W_j z_j + c \quad y = \frac{1}{1 + e^{-\hat{y}}} \quad Y = \text{round}(y)$$

Ztrátová funkce pro klasifikaci: kros-entropie

$$\mathcal{L}(Y(x; \mathbf{w}, \mathbf{W}, \mathbf{b}, c), Y^*) = - \sum_x p(x) \log(q(x))$$

MNIST: benchmark strojového učení (*1998)

Modified National Institute of Standards and Technology database, <http://yann.lecun.com/exdb/mnist/>

Sady obrázků o velikosti 28x28 pixelů, rozdělených do 10 kategorií



TensorFlow (*2017)

Open-source knihovna vyvíjená v Google Brain, umožňuje výpočty na CPU, GPU i TPU

Neuronová síť jako **výpočetní graf**

Vstupem modelu bývá v každém kroku vybraná sada (minibatch) příkladů → mnoharozměrná pole → „tensory“

Součástí TF je i vysokoúrovňové rozhraní **Keras**

<https://www.tensorflow.org>

Příklad klasifikace: https://www.tensorflow.org/tutorials/keras/basic_classification

