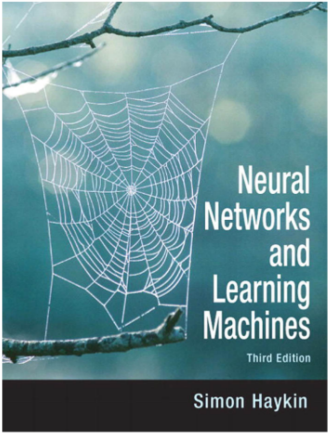
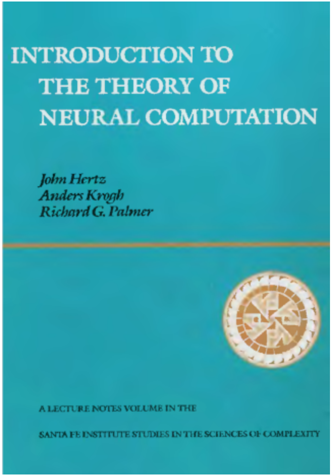
Sztuczne sieci neuronowe

Marcin Jukiewicz   Andrzej Gajda

Sztuczna Inteligencja

22.05.2019


Neural Networks and Learning Machines
Third Edition
Simon Haykin

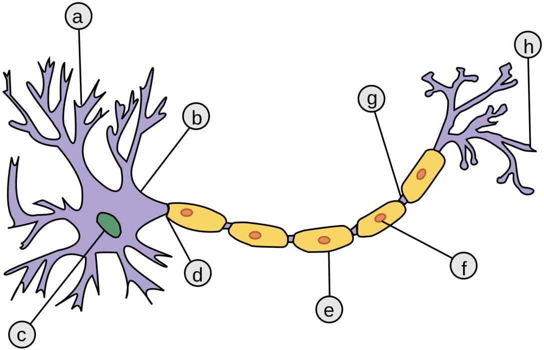

INTRODUCTION TO THE THEORY OF NEURAL COMPUTATION
John Hertz
Anders Krogh
Richard G. Palmer
A LECTURE NOTES VOLUME IN THE
SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY

- klasyfikacja
- tworzenie wzorców
- rozpoznawanie wzorców
- aproksymacja funkcji

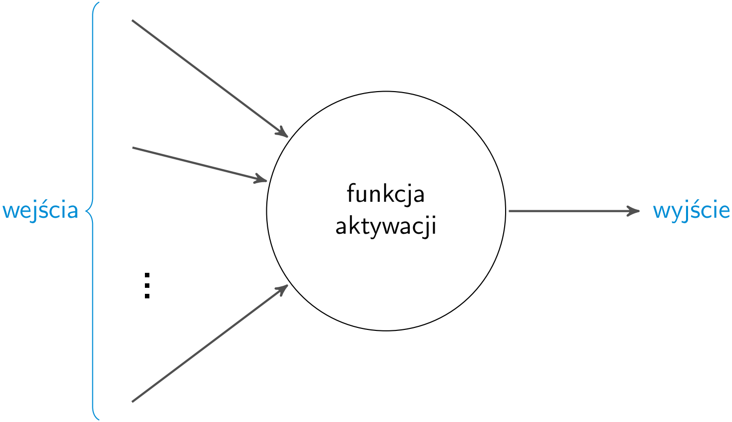
The history of these sorts of ideas in psychology originates with Aristotle. Yet as a basis for computational or neural modelling we can trace them to the paper of McCulloch and Pitts [1943]. . . (Hertz i in., 1990; s. 6)

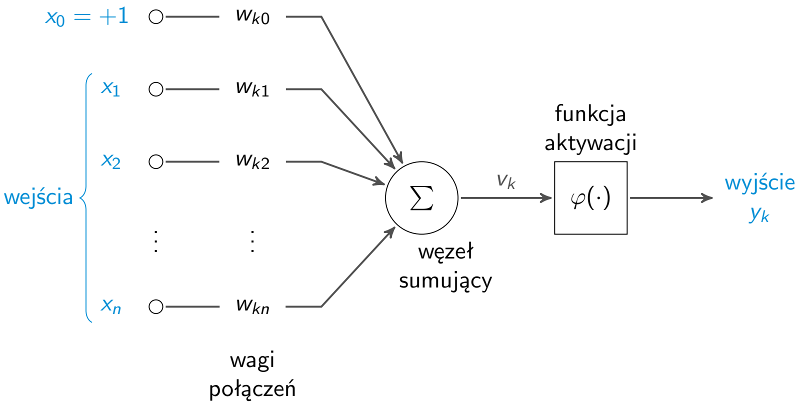
- 1943 — McCulloch i Pitts: pierwszy artykuł
- $\sim$1960 — Rosenblatt: *perceptron*
- 1974 — Werbose: *backpropagation*

Neuron biologiczny

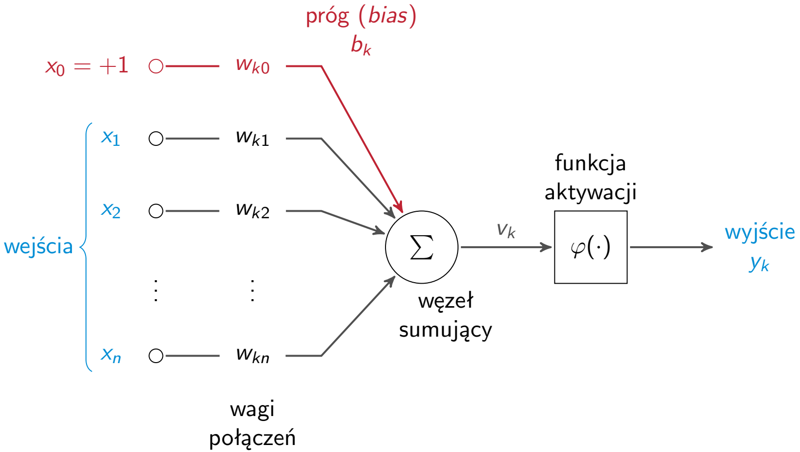


Źródło: `https://pl.wikipedia.org/wiki/Neuron`


wejścia
funkcja aktywacji
wyjście



$x_0 = +1$
w_{k0}
x_1
w_{k1}
x_2
w_{k2}
wejścia
$\vdots$
$\vdots$
x_n
w_{kn}
wagi
połączeń
Σ
węzeł
sumujący
v_k
funkcja
aktywacji
$\varphi(\cdot)$
wyjście
y_k


próg (*bias*)
b_k
$x_0 = +1$
w_{k0}
x_1
w_{k1}
x_2
w_{k2}
wejścia
x_n
w_{kn}
Σ
v_k
funkcja
aktywacji
$\varphi(\cdot)$
wyjście
y_k
węzeł
sumujący
wagi
połączeń

Neuron

$$y_k = \varphi(v_k)$$

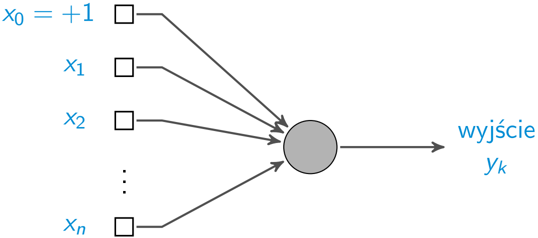
$$v_k = u_k + b_k$$

$$u_k = \sum_{i=1}^{n} w_{ki} x_i$$

$$b_k = x_0 w_{k0}$$

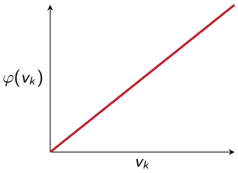
$$y_k = \varphi\left(\sum_{i=1}^{n} w_{ki} x_i + x_0 w_{k0}\right)$$

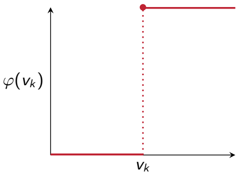
- y_k — wartość wyjściowa (*output*)
- $\varphi(\cdot)$ — funkcja aktywacji (*activation function*)
- v_k — potencjał aktywacji (*activation potential*, *induced local field*)
- u_k — wyjście węzła sumującego (*linear combiner output*)
- $x_1, \ldots, x_n$ — sygnały wejściowe (*input signals*)
- $w_{k1}, \ldots, w_{kn}$ — wagi połączeń (*synaptic weights*)
- b_k — próg aktywacji (*bias*)

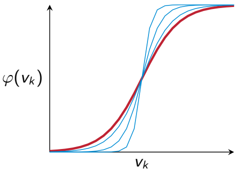


$$y_k = \varphi\left(v_k\right)$$

Funkcja aktywacji ($\varphi(\cdot)$)







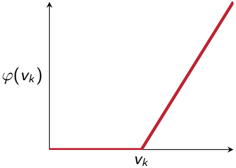
$$\varphi(v_k) = v_k$$

$$\varphi(v_k) = \begin{cases} 1 & \text{if } v_k \geqslant 0 \\ 0 & \text{if } v_k < 0 \end{cases}$$

$$\varphi(v_k) = \frac{1}{1 + \exp(-\beta v_k)}$$

- Relu:

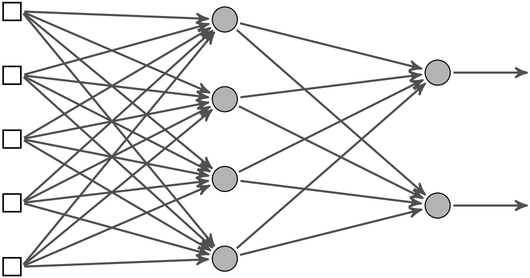
$$\varphi(v_k) = \max(0, v_k)$$



- Softmax:

$$\varphi(v_k) = \frac{\exp(v_k)}{\sum_{j=1}^{n} \exp(v_j)}$$

Perceptron



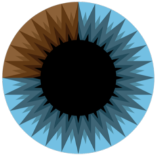
warstwa wejściowa (*input layer*)

warstwa neuronów ukrytych (*hidden layer*)

warstwa neuronów wyjściowych (*output layer*)

Czego potrzebujemy

- Pochodne (*derivatives*)
  - oznaczenie: $\frac{dy}{dx}$, $y'(x)$
  - pochodna cząstkowa (*partial derivative*)
    oznaczenie: $\frac{\partial y}{\partial x}$
- Wektory (*vectors*)
  - oznaczenie: $\mathbf{a}$, $\overline{a}$, $\overrightarrow{AB}$
  - wektory jednostowe / wersory (*unit vectors*): $\hat{\imath}$, $\hat{\jmath}\dots$
- Macierze (*matrices*)
  - oznaczenie: $\mathbf{A}$, $\mathbf{W}\dots$
  - mnożenie, dodawanie macierzy
- Gradient
  - oznaczenie: $\nabla f(x_1, \dots, x_n) = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right)$



3Blue1Brown

$$X = \{\overline{x}_1, \overline{x}_2, \dots, \overline{x}_n\}$$
$$Y = \{\overline{y}_1, \overline{y}_2, \dots, \overline{y}_n\}$$
$$O = \{\overline{o}_1, \overline{o}_2, \dots, \overline{o}_n\}$$

gdzie $\overline{x}_i \in \mathbb{R}^s$, $\overline{y}_i \in \mathbb{R}^t$, $\overline{o}_i \in \mathbb{R}^t$ dla $i \in [1, n]$

- *Mean squared error / quadratic cost / maximum likelihood / sum squared error*:

$$C_{MSE}(y_i, o_i) = \frac{1}{n} \sum_{j=1}^{n} (o_j - y_j)^2$$

$$\nabla_{y_j} C_{MSE} = -2 \cdot (o_j - y_j)$$

$$X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$$
$$Y = \{\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n\}$$
$$O = \{\overline{o}_1, \overline{o}_2, \ldots, \overline{o}_n\}$$

gdzie $\overline{x}_i \in \mathbb{R}^s$, $\overline{y}_i \in \mathbb{R}^t$, $\overline{o}_i \in \mathbb{R}^t$ dla $i \in [1, n]$

- *Exponential cost*:

$$C_{EXP}(\overline{y}_i, \overline{o}_i) = \tau \cdot \exp\left(\frac{1}{\tau}\sum_{j=1}^{t}(o_j - y_j)^2\right)$$

$$\nabla_{y_j} C_{EXP} = \frac{2}{\tau}(o_j - y_j) \cdot C_{EXP}$$

$$X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$$
$$Y = \{\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n\}$$
$$O = \{\overline{o}_1, \overline{o}_2, \ldots, \overline{o}_n\}$$

gdzie $\overline{x}_i \in \mathbb{R}^s$, $\overline{y}_i \in \mathbb{R}^t$, $\overline{o}_i \in \mathbb{R}^t$ dla $i \in [1, n]$

- *Cross-entropy cost*:

$$C_{CE}(\overline{y}_i, \overline{o}_i) = -\frac{1}{n} \cdot \sum_{j=1}^{t} [y_j \ln(o_j) + (1 - y_j) \ln(1 - o_j)]$$

$$\nabla_{y_j} C_{CE} = \frac{o_j - y_j}{o_j \cdot (1 - o_j)}$$

$$\Delta W = -\eta \cdot \nabla_W$$

$$\Delta W = -\eta \cdot \nabla_{W(i)} + \alpha \Delta W(i-1)$$

- *Backpropagation* z momentem i „tarciem"

$$w_t = w_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\delta w_t \qquad v_t = \beta_2 m_{t-1} + (1-\beta_2)(\delta w_t)^2$$

- Stochastyczny (*stochastic*)
- *Mini-batch*
- *Batch* (wsad, partia, pakiet)

- Iteracja (*iteration*) — przejście sygnału przez sieć oraz propagacja wsteczna błędu (*forward pass*, *backward pass*)
- *Epoch* — iteracja **wszystkich** przykładów uczących

- Nieliniowość (*nonlinearity*)
- Mapowanie wejście-wyjście (*input-output mapping*)
- Adaptacyjność (*adaptivity*)
- „Uzasadnienie” odpowiedzi (*evidential response*)
- Zachowanie kontekstu (*contextual information*)
- Odporność na wady (*fault tolerance*)
- *VLSI implementability* (*Very Large Scale Integrated technology*)
- Jednorodność struktury (*uniformity of analysis and design*)
- Analogia neurobilogiczna (*neurobiological analogy*)

- Duża ilość przykładów uczących
  - czas
  - moc obliczeniowa
- Ograniczone możliwości „zaprojektowania” sztucznej sieci neuronowej
- *Black box*

Dziękujemy za uwagę