# Uczenie sieci neuronowych

Lato 2025

Kacper Borys

# Spis treści

# Definicje

### Definicja - perceptron

Niech $\vec{x} \in \mathbb{R}^n$. Perceptronem nazywamy funkcję $P(\vec{x}) = F(\vec{x} \cdot \vec{w} + b)$, gdzie F - funkcja aktywacji (np. sigmoidalna, ReLU, tanh), $\vec{w} \in \mathbb{R}^n, b \in \mathbb{R}$.

Przykładowo - mamy wektor $\vec{x} = (1, 7, 3)$, wagi $\vec{w} = (0.1, 0.5, -2)$ oraz b = 1. Dla tych wartości $P(\vec{x}) = \max(0, -1.4) = 0$.
Oznacza to, że dla takich danych perceptron nie aktywuje się.

## Definicja - warstwa

Warstwą nazywamy funkcję $W(\vec{x}) = (P_1(\vec{x}), P_2(\vec{x}), ..., P_k(\vec{x}))$, gdzie $P_i$ to $i$-ty perceptron i $k \in \mathbb{N}^+$ to liczba perceptronów w warstwie.

Kontynuując przykład. Mamy $\vec{x} = (1, 7, 3)$, wtedy dla konkretnej warstwy $W(\vec{x}) = (0, 0.01, 65.4, 3.1, \cdots, 0.422)$. Jest to nasz nowy wektor danych który możemy wykorzystać jako $\vec{x}'$ w następnej warstwie.

Równoważnie możemy przedstawić naszą warstwę $W(\vec{x})$ jako $F\left(\mathcal{W}\vec{x} + \vec{b}\right)$,

gdzie $\mathcal{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \cdots & w_{k,n} \end{pmatrix}$ to macierz wag, a $\vec{b} = (b_1, b_2, \cdots, b_k)$ to wektor przesunię-cia.

# Sieć neuronowa



## Definicja - sieć neuronowa

Siecią neuronową nazywamy funkcję $N(\vec{x}) = \vec{y}$, gdzie $\vec{x} \in \mathbb{R}^n$ oraz $\vec{y} \in \mathbb{R}^m, m, n \in \mathbb{N}^+$. Jest ona złożeniem funkcji $W^{\text{in}}(\vec{x})$, $W_n^{\text{h}}\left(\vec{h}_{n-1}\right)$ oraz $W^{\text{out}}\left(\vec{h}_n\right)$, gdzie $\vec{h}_0 = W^{\text{in}}(\vec{x})$. Co za tym idzie $N(\vec{x}) = W^{\text{out}}\left(W_k^h\left(W_{k-1}^h\left(\cdots\left(W_1^h\left(W^{\text{in}}(\vec{x})\right)\right)\right)\right)\right)$

# Gradient



**Definicja - gradient**

Gradientem funkcji $f$ w punkcie $x$ nazywamy wektor $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right)$.

Mając jakąś funkcję która przyjmuje wektor, możemy zastosować na niej gradient - zróżniczkować ją względem każdego z parametrów. W ten sposób uzyskujemy wektor, który pokazuje w którą stronę powinniśmy iść aby zwiększyć wartość funkcji.

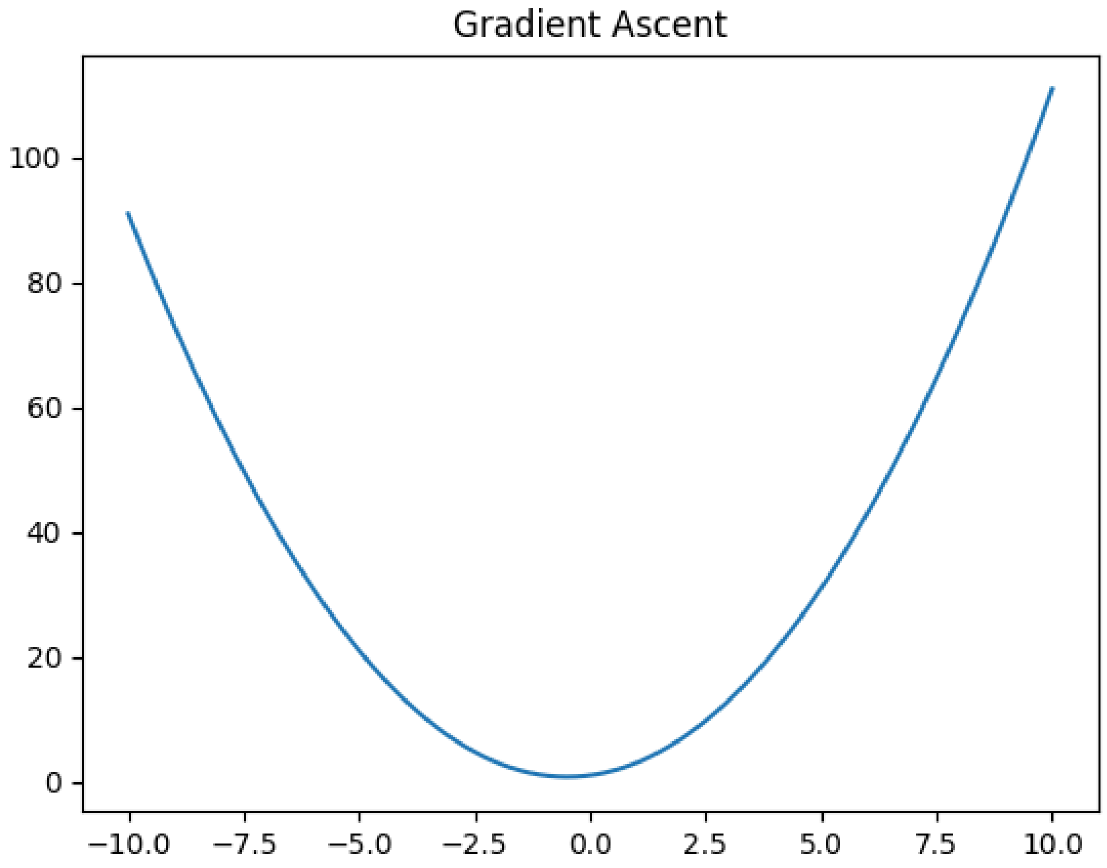
Przykładowo - mamy funkcję $f(x) = x^2 + 2x + 1$, wtedy $\nabla f(x) = (2x + 2)$, a dla $x = 1$ mamy $\nabla f(1) = 4$. Czyli jeśli jesteśmy w $x = 1$ to powinniśmy iść w prawo na osi liczbowej.

# Gradient



**Uwaga**

Gradient wskazuje **kierunek** w którym powinniśmy się poruszać. Długość wektora obrazuje **szybkość** zmiany funkcji.


Gradient Ascent
100
80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

# Uczenie sieci

Niech $p(x) = ax^2 + bx + c$. Aby uzyskać wartość $p(1) = a + b + c = 2$ musimy dobrać odpowiednie parametry. Możemy wyliczyć je analitycznie, ale możemy również wykorzystać gradient.

# Przykład - wstęp



Skoro gradient wskazuje kierunek największego wzrostu funkcji, to możemy iść w przeciwną stronę aby zmniejszyć wartość funkcji, innymi słowy - minimalizować ją.

To co chcemy minimalizować to $L(a, b, c) = (2 - p(a, b, c))^2$ - nazywamy to funkcją kosztu. Pokazuje nam ona jak bardzo funkcja oddalona jest od zadanej wartości.

Przykładowo - dobierzmy parametry $a = 1, b = 1, c = 0$, wtedy $p(1) = 2$, a funkcja kosztu wynosi 0. Jeśli dobierzemy $a = 1, b = 1, c = 1$, wtedy $p(1) = 3$, a funkcja kosztu wynosi 1.

Gdy nałożymy gradient na funkcję kosztu i zmieniemy znak otrzymamy wektor który wskazuje na największy spadek funkcji.

# Przykład - gradient



Obliczmy gradient funkcji kosztu.

$$\nabla L(a, b, c) = \left(\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial c}\right)$$

Korzystając z reguły łańcuchowej mamy:

$$\nabla L(a, b, c) = 2(p(a, b, c) - 2)(\nabla p(a, b, c))$$

# Przykład - gradient decent

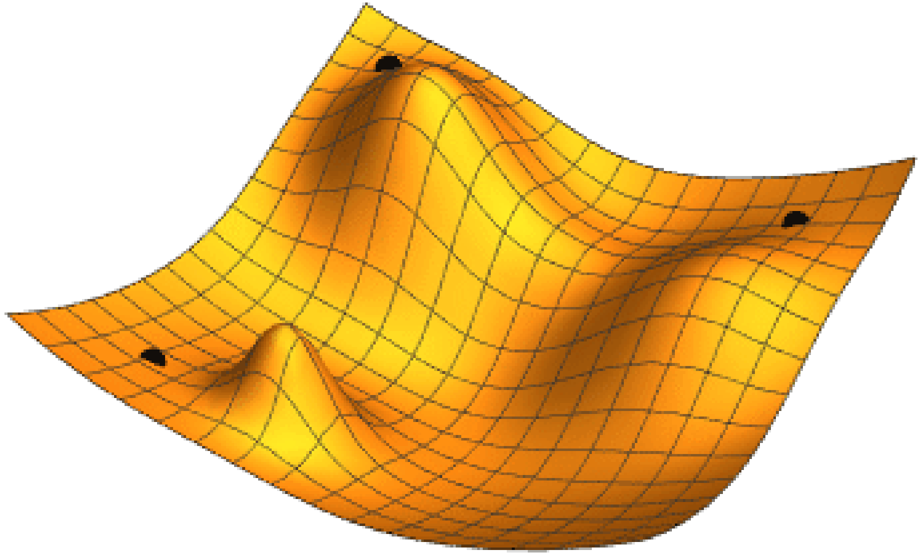


Skoro mamy gradient, to możemy go wykorzystać do zmiany parametrów. Zbudujmy algorytm który będzie aktualizował nasze parametry:

zainicuj parametry $a, b, c$
zainicjuj krok $\alpha$
zainicjuj $\varepsilon$
**dopóki** $L(a, b, c) > \varepsilon$
    oblicz gradient $\nabla L(a, b, c)$
    (a, b, c) = (a, b, c) - $\alpha * \nabla L(a, b, c)$
**koniec**

# Przykład - gradient decent





# Propagacja wsteczna

## Definicja - propagacja wsteczna

Propagacją wsteczną nazywamy proces aktualizowania parametrów sieci neuronowej przy pomocy $\nabla L(\vec{y} - N(\vec{p}))$ gdzie $\vec{p}$ jest wektorem złożonym z wag oraz biasów.

Znając gradient $W^{\text{out}}$, dzięki regule łańcuchowej, możemy obliczyć gradient $W_n^{\text{h}}$, a znając gradient $W_n^{\text{h}}$ możemy obliczyć gradient $W_{n-1}^{\text{h}}$ i tak dalej aż do $W^{\text{in}}$. Tym sposobem możemy uczyć naszą sieć neuronową.

# Propagacja wsteczna



Te równania dokładnie opisują algorytm propagacji wstecznej:

$$\begin{aligned}
\delta^{\text{out}} &= \nabla_{\sigma(\vec{h})} L \odot \sigma'\left(\vec{h}\right) \\
\delta^{\text{h}}_{n-1} &= (W_n)^T \delta_n \odot \sigma'\left(\vec{h}_n\right) \\
\delta_n &= \frac{\partial L}{\partial \vec{b}_n} \\
\sigma\left(\vec{h}_{n-1}\right)\delta_n &= \frac{\partial L}{\partial \vec{w}_n}
\end{aligned}$$

Gdzie $\odot$ jest mnożeniem Hadamarda (*element-wise*), $\sigma$ jest funkcją aktywacji, a $L$ jest funkcją kosztu.