

zad06-activation-functions

November 17, 2025

1 Zadanie 06: Funkcje Aktywacji - ReLU

1.1 Wariant: 3 - Funkcja ReLU (Rectified Linear Unit)

1.1.1 Cel

Przeprowadzić badania funkcji aktywacji ReLU zgodnie z wariantem zadania: 1. Obliczyć gradient funkcji 2. Wyświetlić funkcję wraz z gradientem na jednym wykresie 3. Opisać zagadnienia w których używają daną funkcję aktywacji

1.1.2 Funkcja ReLU

Funkcja ReLU (Rectified Linear Unit) jest jedną z najpopularniejszych funkcji aktywacji w głębokim uczeniu się.

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{jeśli } x > 0 \\ 0 & \text{jeśli } x \leq 0 \end{cases}$$

Gradient funkcji ReLU:

$$\text{ReLU}'(x) = \begin{cases} 1 & \text{jeśli } x > 0 \\ 0 & \text{jeśli } x \leq 0 \end{cases}$$

```
[5]: import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams

rcParams['figure.figsize'] = (14, 5)
rcParams['font.size'] = 12

x = np.linspace(-5, 5, 1000)

# Funkcja ReLU
def relu(x):
    """Funkcja aktywacji ReLU"""
    return np.maximum(0, x)

# Gradient ReLU
def relu_gradient(x):
```

```

"""Gradient funkcji ReLU"""
return np.where(x > 0, 1, 0)

# Obliczenie wartości
y_relu = relu(x)
y_gradient = relu_gradient(x)

print("Funkcja ReLU (Rectified Linear Unit)")
print(f"Definicja: ReLU(x) = max(0, x)")
print(f"Gradient: ReLU'(x) = 1 jeśli x > 0, 0 jeśli x <= 0")

```

Funkcja ReLU (Rectified Linear Unit)
 Definicja: $\text{ReLU}(x) = \max(0, x)$
 Gradient: $\text{ReLU}'(x) = 1 \text{ jeśli } x > 0, 0 \text{ jeśli } x \leq 0$

```

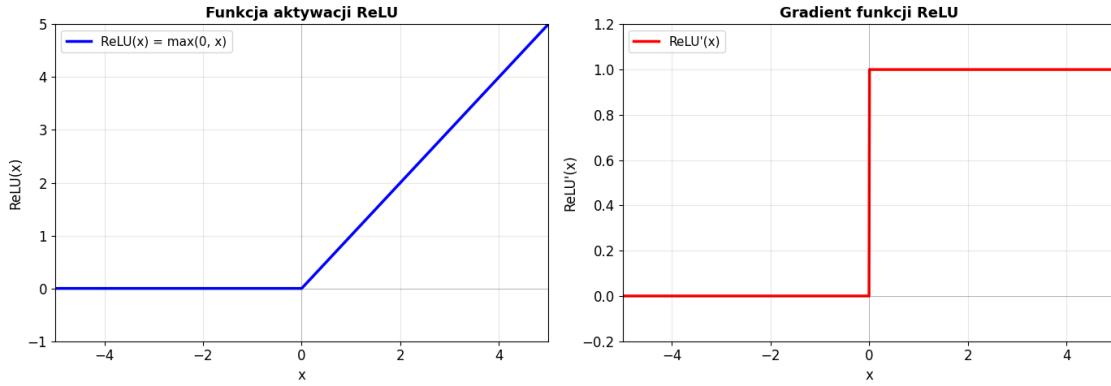
[6]: # Wykres funkcji i gradientu
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Podwykres 1: Funkcja ReLU
ax1.plot(x, y_relu, 'b-', linewidth=2.5, label='ReLU(x) = max(0, x)')
ax1.axhline(y=0, color='k', linestyle='--', linewidth=0.5, alpha=0.3)
ax1.axvline(x=0, color='k', linestyle='--', linewidth=0.5, alpha=0.3)
ax1.grid(True, alpha=0.3)
ax1.set_xlabel('x', fontsize=12)
ax1.set_ylabel('ReLU(x)', fontsize=12)
ax1.set_title('Funkcja aktywacji ReLU', fontsize=13, fontweight='bold')
ax1.legend(fontsize=11, loc='upper left')
ax1.set_xlim(-5, 5)
ax1.set_ylim(-1, 5)

# Podwykres 2: Gradient ReLU
ax2.plot(x, y_gradient, 'r-', linewidth=2.5, label="ReLU'(x)")
ax2.axhline(y=0, color='k', linestyle='--', linewidth=0.5, alpha=0.3)
ax2.axvline(x=0, color='k', linestyle='--', linewidth=0.5, alpha=0.3)
ax2.grid(True, alpha=0.3)
ax2.set_xlabel('x', fontsize=12)
ax2.set_ylabel("ReLU'(x)", fontsize=12)
ax2.set_title('Gradient funkcji ReLU', fontsize=13, fontweight='bold')
ax2.legend(fontsize=11, loc='upper left')
ax2.set_xlim(-5, 5)
ax2.set_ylim(-0.2, 1.2)

plt.tight_layout()
plt.show()

```



```
[7]: # Połączony wykres: funkcja i gradient na jednym wykresie
fig, ax = plt.subplots(figsize=(12, 6))

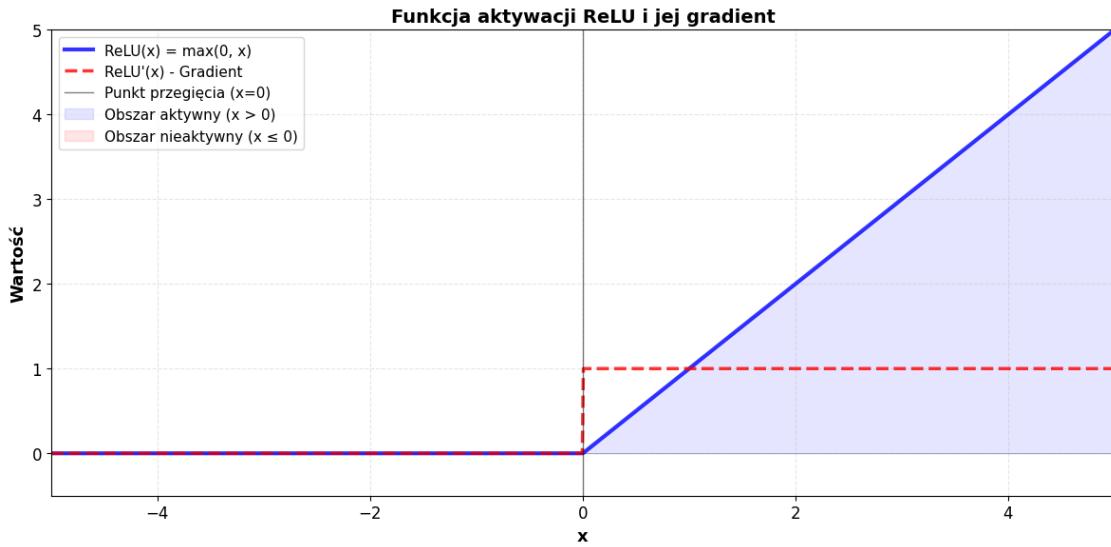
# Normalizacja gradientu aby był widoczny obok funkcji
ax.plot(x, y_relu, 'b-', linewidth=3, label='ReLU(x) = max(0, x)', alpha=0.8)
ax.plot(x, y_gradient, 'r--', linewidth=2.5, label="ReLU'(x) - Gradient", alpha=0.8)

ax.axhline(y=0, color='k', linestyle='-', linewidth=0.5, alpha=0.3)
ax.axvline(x=0, color='k', linestyle='-', linewidth=1, alpha=0.5, label='Punkt\u2022 przegięcia (x=0)')

ax.fill_between(x, 0, y_relu, where=(x >= 0), alpha=0.1, color='blue', label='Obszar aktywny (x > 0)')
ax.fill_between(x, 0, y_relu, where=(x < 0), alpha=0.1, color='red', label='Obszar nieaktywny (x < 0)')

ax.grid(True, alpha=0.3, linestyle='--')
ax.set_xlabel('x', fontsize=13, fontweight='bold')
ax.set_ylabel('Wartość', fontsize=13, fontweight='bold')
ax.set_title('Funkcja aktywacji ReLU i jej gradient', fontsize=14, fontweight='bold')
ax.legend(fontsize=11, loc='upper left')
ax.set_xlim(-5, 5)
ax.set_ylim(-0.5, 5)

plt.tight_layout()
plt.show()
```



```
[8]: print("ANALIZA NUMERYCZNA FUNKCJI ReLU")

# Wartości w charakterystycznych punktach
test_points = [-2, -1, -0.1, 0, 0.1, 1, 2]

print(f"\n{'x':>10} | {'ReLU(x)':>15} | {'ReLU'(x):>15}")
print("-" * 45)
for x_val in test_points:
    relu_val = relu(np.array([x_val]))[0]
    grad_val = relu_gradient(np.array([x_val]))[0]
    print(f"{x_val:>10.1f} | {relu_val:>15.6f} | {grad_val:>15.6f}")

print("WŁAŚCIWOŚCI FUNKCJI ReLU")
print("1. Dziedzina: (-∞, +∞)")
print("2. Zakres: [0, +∞]")
print("3. Punkt przegięcia: x = 0")
print("4. Gradient dla x > 0: 1.0 (stały)")
print("5. Gradient dla x < 0: 0.0 (brak gradientu)")
print("6. Ciągłość: Funkcja ciągła, ale nieróżniczkowalna w x=0")
```

ANALIZA NUMERYCZNA FUNKCJI ReLU

x	ReLU(x)	ReLU'(x)
-2.0	0.000000	0.000000
-1.0	0.000000	0.000000
-0.1	0.000000	0.000000
0.0	0.000000	0.000000
0.1	0.100000	1.000000

1.0	1.000000	1.000000
2.0	2.000000	1.000000

WŁAŚCIWOŚCI FUNKCJI ReLU

1. Dziedzina: $(-\infty, +\infty)$
2. Zakres: $[0, +\infty)$
3. Punkt przegięcia: $x = 0$
4. Gradient dla $x > 0$: 1.0 (stały)
5. Gradient dla $x < 0$: 0.0 (brak gradientu)
6. Ciągłość: Funkcja ciągła, ale nieróżniczkowalna w $x=0$

1.2 Wniosek: Zastosowania funkcji ReLU w sieciach neuronowych

1.2.1 1. Dlaczego funkcja ReLU jest tak popularna?

Funkcja ReLU (Rectified Linear Unit) jest jedną z najczęściej używanych funkcji aktywacji w głębokim uczeniu ze względu na następujące zalety:

- **Prostota obliczeniowa:** Bardzo szybka w obliczaniu (tylko $\max(0, x)$)
- **Efektywność gradientu:** Gradient wynosi 1 dla dodatnich wartości, co ułatwia propagację wsteczną
- **Redukcja znikającego gradientu:** W przeciwieństwie do sigmoid czy tanh, ReLU nie zmniejsza gradientów eksponencjalnie

1.2.2 2. Główne zastosowania:

a) Sieci konwolucyjne (CNN)

- Standard w wielu architekturach (AlexNet, VGG, ResNet)
- Szczególnie efektywna w ekstrakcji cech z obrazów

b) Głębokie sieci neuronowe (DNN)

- Umożliwia trenowanie bardzo głębokich sieci
- Pozwala na wprowadzenie nieliniowości bez znikającego gradientu

c) Przetwarzanie języka naturalnego (NLP)

- Wykorzystywana w warstwach gęsto połączonych transformerów
- Część standardowych implementacji BERT, GPT