

Opracowanie wyników

Grupa zadań 4: Dwuwarstwowa sieć neuronowa dla problemu XOR

Maksymilian Mazur

1 Źródłowy zbiór danych

Zbiór danych składa się z czterech wzorców reprezentujących tablice prawdy operacji XOR:

$x^{(1)}$	$x^{(2)}$	$d \text{ (XOR)}$
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 1: Tablica prawdy operacji XOR

W celu zwiększenia liczności zbioru treningowego, wokół każdego z czterech punktów bazowych wygenerowano równą ilość próbek z dodanym szumem gaussowskim ($\mathcal{N}(0, 0.1)$), ograniczając wartości do przedziału $[0, 1]$. Ostatecznie:

- Zbiór treningowy: 208 próbek (70%)
- Zbiór testowy: 88 próbek (30%)

2 Struktura sieci neuronowej

2.1 Architektura

Zaimplementowano dwuwarstwowa sieć neuronowa:

- **Warstwa wejściowa:** $k = 2$ wejścia ($x^{(1)}, x^{(2)}$)
- **Warstwa ukryta:** $h = 2$ perceptrony z wyjściami ($v^{(1)}, v^{(2)}$)
- **Warstwa wyjściowa:** $r = 1$ perceptron z wyjściem y

Każda warstwa posiada dodatkowe wejście (bias) o wartości stałej równej 1.

2.2 Funkcja aktywacji

We wszystkich perceptronach zastosowano unipolarną funkcję sigmoidalną:

$$f(s) = \frac{1}{1 + e^{-\beta s}} \quad (1)$$

gdzie $\beta = 2$ jest parametrem nachylenia funkcji. Pochodna tej funkcji wynosi:

$$f'(s) = \beta \cdot f(s) \cdot [1 - f(s)] \quad (2)$$

2.3 Funkcja błędu

Jako miarę błędu sieci zastosowano funkcję średniego błędu kwadratowego (MSE):

$$e = \frac{1}{2} \sum_{i=1}^r (d^{(i)} - y^{(i)})^2 = \frac{1}{2} \sum_{i=1}^r (\varepsilon^{(i)})^2 \quad (3)$$

gdzie $d^{(i)}$ oznacza oczekiwane wyjście, $y^{(i)}$ rzeczywiste wyjście sieci, a $\varepsilon^{(i)} = d^{(i)} - y^{(i)}$ błąd predykcji. W celu ułatwienia dalszych obliczeń, MSE jest dzielony przez 2.

3 Postać zbioru uczącego

Zbiór uczący ma postać zbioru par wejście-wyjście:

$$S_N = \{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, (\mathbf{x}_N, d_N)\} \quad (4)$$

gdzie:

- $\mathbf{x}_n = [x_n^{(1)}, x_n^{(2)}]^T \in \mathbb{R}^2$ – wektor wejściowy
- $d_n \in \{0, 1\}$ – oczekiwane wyjście (etykieta)
- $N = 210$ – liczba próbek treningowych

4 Algorytm uczenia – metoda wstecznej propagacji błędu

4.1 Propagacja w przód

Dla danego wektora wejściowego $\mathbf{x} = [x^{(1)}, x^{(2)}]^T$ obliczenia przebiegają następująco:

Warstwa ukryta:

$$s_i = \sum_{j=1}^k w_i^{(j)} x^{(j)} + w_i^{(0)}, \quad i = 1, \dots, h \quad (5)$$

$$v^{(i)} = f(s_i) = \frac{1}{1 + e^{-\beta s_i}} \quad (6)$$

Warstwa wyjściowa:

$$s_{h+l} = \sum_{i=1}^h w_{h+l}^{(i)} v^{(i)} + w_{h+l}^{(0)}, \quad l = 1, \dots, r \quad (7)$$

$$y^{(l)} = f(s_{h+l}) = \frac{1}{1 + e^{-\beta s_{h+l}}} \quad (8)$$

4.2 Propagacja wsteczna

4.2.1 Warstwa wyjściowa

Obliczenie gradientu błędu po wagach warstwy wyjściowej:

$$\frac{\partial e}{\partial w_{h+l}^{(i)}} = \frac{\partial e}{\partial y^{(l)}} \cdot \frac{\partial y^{(l)}}{\partial s_{h+l}} \cdot \frac{\partial s_{h+l}}{\partial w_{h+l}^{(i)}} \quad (9)$$

Wprowadzając oznaczenie δ_{h+l} (gradient błędu po sumie ważonej):

$$\delta_{h+l} = -\varepsilon^{(l)} \cdot f'(s_{h+l}) = -(d^{(l)} - y^{(l)}) \cdot \beta \cdot y^{(l)} \cdot (1 - y^{(l)}) \quad (10)$$

Gradienty dla wag i biasu:

$$\frac{\partial e}{\partial w_{h+l}^{(i)}} = \delta_{h+l} \cdot v^{(i)}, \quad i = 1, \dots, h \quad (11)$$

$$\frac{\partial e}{\partial w_{h+l}^{(0)}} = \delta_{h+l} \quad (12)$$

4.2.2 Warstwa ukryta

Gradient błędu propagowany wstecz przez wagi:

$$\frac{\partial e}{\partial w_i^{(j)}} = \left[\sum_{m=1}^r \frac{\partial e}{\partial y^{(m)}} \cdot \frac{\partial y^{(m)}}{\partial s_{h+m}} \cdot \frac{\partial s_{h+m}}{\partial v^{(i)}} \right] \cdot \frac{\partial v^{(i)}}{\partial s_i} \cdot \frac{\partial s_i}{\partial w_i^{(j)}} \quad (13)$$

Definiując δ_i dla warstwy ukrytej:

$$\delta_i = \left[\sum_{m=1}^r \delta_{h+m} \cdot w_{h+m}^{(i)} \right] \cdot f'(s_i) = \left[\sum_{m=1}^r \delta_{h+m} \cdot w_{h+m}^{(i)} \right] \cdot \beta \cdot v^{(i)} \cdot (1 - v^{(i)}) \quad (14)$$

Gradienty dla wag i biasu:

$$\frac{\partial e}{\partial w_i^{(j)}} = \delta_i \cdot x^{(j)}, \quad j = 1, \dots, k \quad (15)$$

$$\frac{\partial e}{\partial w_i^{(0)}} = \delta_i \quad (16)$$

4.3 Aktualizacja wag

Wagi modyfikowane sa metoda gradientu prostego (gradient descent):

$$w^{(n+1)} = w^{(n)} - \gamma \cdot \frac{\partial e}{\partial w} \bigg|_{\mathbf{x}=\mathbf{x}_n, d=d_n} \quad (17)$$

gdzie $\gamma = 0.01$ jest stałym współczynnikiem uczenia (learning rate).

4.4 Notacja macierzowa

W implementacji wykorzystano notacje macierzowa dla zwiększenia efektywności:

Warstwa wyjściowa:

$$\delta_y = -\varepsilon \odot f'(\mathbf{s}_y) \quad (18)$$

$$\frac{\partial e}{\partial \mathbf{W}_y} = \delta_y \mathbf{v}^T \quad (19)$$

$$\frac{\partial e}{\partial \mathbf{w}_y^{(0)}} = \delta_y \quad (20)$$

Warstwa ukryta:

$$\delta_v = (\mathbf{W}_y^T \delta_y) \odot f'(\mathbf{s}_v) \quad (21)$$

$$\frac{\partial e}{\partial \mathbf{W}_v} = \delta_v \mathbf{x}^T \quad (22)$$

$$\frac{\partial e}{\partial \mathbf{w}_v^{(0)}} = \delta_v \quad (23)$$

gdzie \odot oznacza iloczyn Hadamarda.

5 Parametry uczenia

- Współczynnik uczenia: $\gamma = 0.01$
- Parametr funkcji sigmoid: $\beta = 2$
- Liczba epok: 125

6 Uzyskane wyniki

6.1 Proces uczenia

Wartość funkcji błędu MSE na zbiorze treningowym systematycznie malała w kolejnych epokach:

Epoka	MSE
1	0.1295
10	0.1230
101	0.0847
120	0.0676
125	0.0619

Tabela 2: Spadek błędu w wybranych epokach

6.2 Wagi końcowe

Po zakończeniu treningu uzyskano następujące wagi:

Warstwa ukryta:

- Neuron 1: $w_1^{(1)} = -0.707$, $w_1^{(2)} = 1.276$, $w_1^{(0)} = 0.094$
- Neuron 2: $w_2^{(1)} = 2.340$, $w_2^{(2)} = -2.191$, $w_2^{(0)} = 1.243$

Warstwa wyjściowa:

- Neuron 3: $w_3^{(1)} = -1.329$, $w_3^{(2)} = -1.796$, $w_3^{(0)} = 2.194$

6.3 Metryki

- **Dokładność na zbiorze testowym:** 97.73% (86/88 poprawnych predykcji)
- **MSE na zbiorze treningowym (w ostatniej epoce):** 0.0619