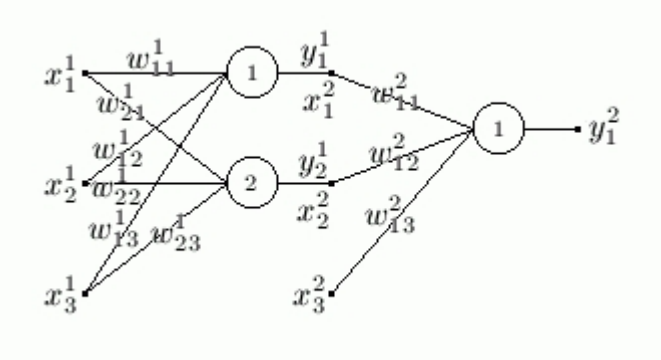
3. Dwuwarstwowa sieć neuronowa

W tym ćwiczeniu zajmiemy się najprostszą z sieci dwuwarstwowych, która rozwiązuje wcale nie banalny problem dwuwymiarowego zadania klasyfikacji XOR. Funkcja XOR zdefiniowana jest w następujący sposób:

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Tradycyjna sieć jednowarstwowa nie jest w stanie rozwiązać tego zadania. Potrafi to już jednak prosta sieć wielowarstwowa mająca 2 warstwy oraz 2 neurony w warstwie wejściowej (pierwszej) i 1 neuron w warstwie wyjściowej (drugiej) (patrz rysunek 1).



Rysunek 1: Model sieci dwuwarstwowej realizującej funkcję XOR.

Do uczenia takiej sieci znakomicie nadaje się algorytm z ćwiczenia 1 po wprowadzeniu pewnych poprawek. Są to jednak zmiany czysto kosmetyczne, prowadzące do uogólnienia na sieci wielowarstwowe a nie jakościowo nowa metoda.

Zanim przejdziemy do ich przedstawienia jeszcze mała uwaga. Otóż teraz do oznaczenia wagi używać będziemy aż trzech indeksów. Zapis w_{bc}^a oznacza, że mówimy o wadze z warstwy a łączącej neuron o numerze c z warstwy $a-1$ z neuronem o numerze b z warstwy a.

Postąpimy podobnie jak we wspomnianym już parokrotnie ćwiczeniu. Chcąc zastosować metodę gradientową musimy policzyć pochodną funkcji celu po kolejnych wagach. Jak się przekonamy wzór na zmianę wag warstwy wyjściowej pozostanie bez zmian. Wzór na zmianę wag w warstwie wcześniejszej uwzględniać będzie natomiast sygnał błędu, nazwyany **sygnałem delta**. Wynika to stąd, że dla wszystkich warstw z wyjątkiem wyjściowej nie znamy prawidłowej odpowiedzi sieci, znajomość której wymagana jest w algorytmie. W związku z tym, wychodząc ze słusznego założenia,

że neuroy z warstw poprzednich mają wpływ na na błąd w warstwie wyjściowej, będziemy ich błąd obliczać w oparciu o błąd warstwy wyjściowej (dokładniej mówiąc, to błąd warstwy I będzie obliczany na podstawie błędu warstwy $I+1$).

Oto jak będzie przebiegało wyprowadzenie potrzebnych wzorów. Przyjmujemy analogiczną postać funkcji błędu

$$E(w) = \frac{1}{2}[t - f(net_1^2)]^2,$$

gdzie net_1^2 to pobudzenie neuronu 1 z warstwy 2.

Dla warstwy wyjściowej otrzymujemy

$$\frac{\partial E}{\partial w_{1p}^2} = \frac{\partial E}{\partial net_1^2}\frac{\partial net_1^2}{\partial w_{1p}^2} = \frac{1}{2}\left([t - f(net_1^2)]^2\right)'_{w_{1p}^2} = \left[t - f(\sum_{k=1}^{1} x_k^2 w_{1k}^2)\right]\left[t - f(\sum_{k=1}^{1} x_k^2 w_{1k}^2)\right]'_{w_{1p}^2} = ...$$

gdzie:

- p zmienia się od 1 do ilości wejść dla warstwy 2, w naszym przypadku do 3;
- suma po k jest od 1 do 3, gdyż sieć ma 2 wejścia +1 sygnał stały, razem 3;
- x_k^2 oznacza $k-ty$ sygnał wejściowy dla warstwy 2;
- w_{1k}^2 oznacza wagę łączącą neuron 1 z warstwy 2 z $k-tym$ wejściem dla warstwy 2 (czyli na ogół z $k-tym$ neuronem);

$$... = -(t - f(net_1^2))f'(net_1^2)x_{1p}^2$$

iloczyn

$$(t - f(net_1^2))f'(net_1^2)$$

oznaczamy przez

$$\delta_1^2 = (t - f(net_1^2))f'(net_1^2)$$

(W tym przypadku oznacza to, że jest to sygnał delta dla 1 neuronu z warstwy 2). Oczywiście

$$net_1^2 = \sum_{p=1}^{3} x_p^2 w_{1p}^2$$

gdzie

$$x_1^2 = f(\sum_{m=1}^{3} x_m^1 w_{1m}^1)$$

$$x_2^2 = f(\sum_{m=1}^{3} x_m^1 w_{2m}^1)$$

$$x_3^2 = 1.$$

Teraz zajmijmy się warstwą pierwszą.

$$\frac{\partial E}{\partial w_{qp}^1} = \frac{1}{2}\left([t - f(net_1^2)]^2\right)'_{w_{qp}^1} = \left[t - f(net_1^2)\right]\left[t - f(net_1^2)\right]'_{w_{qp}^1} =$$

$$= -\left[t - y\right] f'(net_1^2)(net_1^2)'_{w_{qp}^1} = -\left[t - y\right] f'(net_1^2)(x_1^2 w_{11}^2 + x_2^2 w_{12}^2 + x_3^2 w_{13}^2)'_{w_{qp}^1} =$$

$$= -\left[t - y\right] f'(net_1^2)\left(f(net_1^1)w_{11}^2 + f(net_2^1)w_{12}^2 + w_{13}^2\right)'_{w_{qp}^1} =$$

$$= -\left[t-y\right] f'(net_1^2) f'(net_q^1) w_{1q}^2 \left(net_q^1\right)'_{w_{qp}^1} =$$

$$= -\left[t-y\right] f'(net_1^2) f'(net_q^1) w_{1q}^2 \left(x_1^1 w_{q1}^1 + x_2^1 w_{q2}^1 + x_3^1 w_{q3}^1\right)'_{w_{qp}^1} =$$

$$= -\left[t-y\right] f'(net_1^2) f'(net_q^1) w_{1q}^2 x_p^1 = -\delta_1^2 f'(net_q^1) w_{1q}^2 x_p^1 = -\delta_q^1 x_p^1$$

gdzie

$$\delta_q^1 = \delta_1^2 f'(net_q^1) w_{1q}^2$$

Jeśli teraz sieć miałaby więcej warstw, to analogiczne obliczenia należałoby przeprowadzić dla kolejnych warstw poprzedzających te dwie, dla których właśnie wyprowadziliśmy odpowiednie wzory.

Algorytm

Definiujemy w następujący sposób kolejne elementy zbioru uczącego:
$p_1 = \{-1, -1, 1\}$
$p_2 = \{-1, 1, 1\}$
$p_3 = \{1, -1, 1\}$
$p_4 = \{1, 1, 1\}$

$t_1 = \{0\}$
$t_2 = \{1\}$
$t_3 = \{1\}$
$t_4 = \{0\}$

1. Wybór $\eta > 0$ (współczynnik uczenia), $E_{max} > 0$ (maksymalny błąd jaki chcemy osiągnąć), $C_{max} > 0$ (ilość kroków uczenia).
2. Losowy wybór początkowych wartości wag jako niewielkich liczb (na przykład z przedziału $[-1, 1]$); $c := 0$.
3. $l := 0$, $E := 0$.
4. Podanie jednego z obrazów ze zbioru P na wejścia sieci.
5. Obliczenie sygnału wyjściowego sieci, czyli y_1^2.
6. Obliczenie sygnałów błędu

$$\delta_1^2 = (t - f(net_1^2)) f'(net_1^2)$$

$$\delta_1^1 = \delta_1^2 w_{11}^2 f'(net_1^1)$$
$$\delta_2^1 = \delta_1^2 w_{12}^2 f'(net_2^1)$$

7. Uaktualnienie wartości wag według wzoru

$$w_{bc}^a = w_{bc}^a + \eta \delta_b^a x_c^a$$

8. Obliczenie błędu
$$E = E + \frac{1}{2}(t - y)^2$$

9. Jeśli $l < ilość\_obrazów$ to $l := l + 1$ i przejście do kroku 4.

10. Jeśli $E < E_{max}$, to kończymy algorytm. Jeśli $c < C_{max}$, to $c := c + 1$ i przechodzimy do kroku 3. W przeciwnym razie kończymy algorytm.

Zadanie

Należy zaimplementować zaprezentowany algorytm dla problemu XOR.