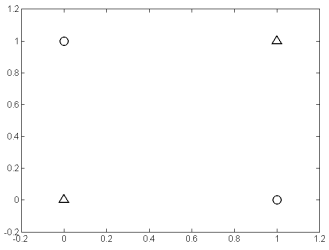
1 Aproksymacja funkcji. Algorytm wstecznej propagacji błędu

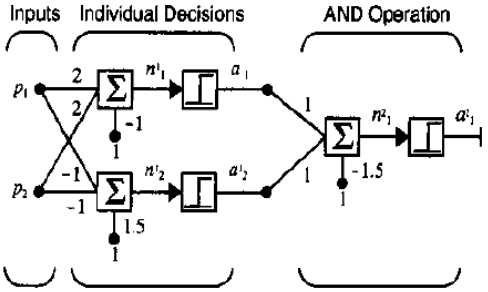
Przedmiot: Sieci Neuronowe

1.1 Wielowarstwowy perceptron

Znamy już ograniczenia jednowarstwowego perceptronu. Wspomniany – na poprzednich zajęciach – problem XOR może być rozwiązany przez sieć dwuwarstwową. Istnieje wiele propozycji topologii wielowarstwowych służących realizacji tego zadania. Przykładowa struktura przedstawiona jest na rys. 1b.



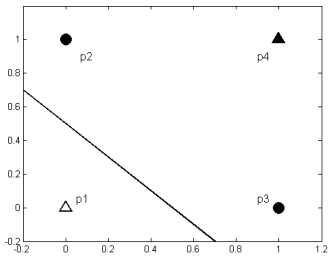
(a) Problem XOR



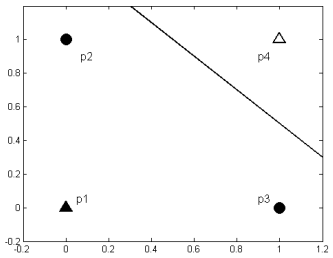
(b) Dwuwarstwowa sieć perceptronowa rozwiązująca problem XOR

Rysunek 1

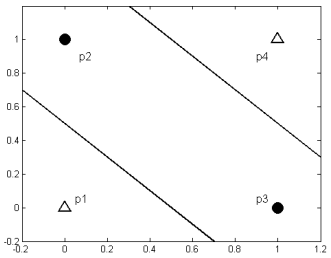
W zaproponowanym rozwiązaniu każdy z neuronów pierwszej warstwy utworzy po dwa obszary decyzyjne. Pierwszy neuron odseparuje próbkę p_1 od pozostałych, natomiast drugi próbkę p_4. Neuron z drugiej warstwy działa jak operator koniunkcji.



(a) Prosta klasyfikacyjna neuronu n_1^1



(b) Prosta klasyfikacyjna neuronu n_2^1



(c) Neuron z drugiej warstwy

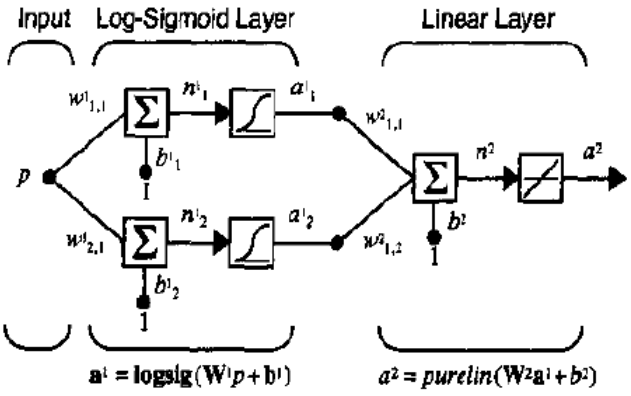
Rysunek 2

1.2 Algorytm wstecznej propagacji błędu

Dotychczas analizowaliśmy sieci w kontekście ich funkcji klasyfikacyjnych. Przedstawimy możliwości wielowarstwowego perceptronu służącego aproksymacji funkcji.
Rozważmy przykład sieci przedstawionej na rys. 3.
W przypadku sieci wielowarstwowej, wyjścia jednej warstwy są wejściami warstwy następnej. Dla naszej sieci, funkcje przejścia neuronów w pierwszej warstwie są typu logarytmiczno sigmoidalnego: $f^1(n) = \frac{1}{1+e^{-n}}$, natomiast w drugiej warstwie liniowego: $f^2(n) = n$. Stąd równanie opisujące działanie sieci:

$$a^2 = purelin(W^2 * (logsig(W^1 * p + b^1)) + b^2) \tag{1}$$

Zmieniając wartości parametrów sieci (wag oraz biasów), modyfikujemy kształt krzywej stanowiącej odpowiedzi sieci. **W celu weryfikacji wpływu parametrów sieci na jej odpowiedzi skorzystaj z *Neural Network***



Rysunek 3: Przykład sieci służącej aproksymacji funkcji (źródło [1])

***Design Demonstration Network Function*:** `nnd11nf`. Sprawdź, jaki skutek przynoszą zmiany wartości wag, a jaki biasu – zwłaszcza z ostatniej warstwy.
Na podstawie analizy przykładu możemy stwierdzić, że dwuwarstwowe perceptrony o sigmoidalnych funkcjach przejścia, w warstwie ukrytej oraz liniowej funkcji w warstwie wyjściowej mogą służyć do aproksymacji dowolnej funkcji, założywszy dostateczną liczbę neuronów w warstwie ukrytej.
W celu przedstawienia algorytmu wstecznej propagacji przeanalizujmy prosty przykład. Zdefiniujmy problem, a dokładnie funkcję, którą będziemy aproksymować przy pomocy naszej sieci. Niech to będzie funkcja kwadratowa określona w przedziale $< -2, 2 >$:
$g(p) = (p-1)^2$ dla $-2 \leqslant p \leqslant 2$
Zainicjujmy parametry sieci małymi, losowymi wartościami, przykładowo:

$$W^1(0) = \begin{bmatrix} 0.09 \\ -0.37 \end{bmatrix}, b^1(0) = \begin{bmatrix} -0.29 \\ -0.17 \end{bmatrix}, W^2(0) = \begin{bmatrix} -0.45 \\ 0.33 \end{bmatrix}, b^2(0) = [0.39].$$

Na programie demonstracyjnym sprawdź jak będzie wyglądać sygnał odpowiedzi sieci, dla jej parametrów o wartościach bliskich zeru.
Teraz możemy przeanalizować algorytm wstecznej propagacji błędu. Obliczmy odpowiedź sieci dla

$$a^0 = p = 1.$$

Wektor wartości odpowiedzi pierwszej warstwy (ukrytej) a^1:

$$a^1 = f^1(W^1 a^0 + b^1) = logsig\left(\begin{bmatrix} 0.09 \\ -0.37 \end{bmatrix} * [1] + \begin{bmatrix} -0.29 \\ -0.17 \end{bmatrix}\right) = logsig\left(\begin{bmatrix} -0.2 \\ -0.54 \end{bmatrix}\right) = \begin{bmatrix} \frac{1}{1+e^{0.2}} \\ \frac{1}{1+e^{0.54}} \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0.37 \end{bmatrix}$$

Natomiast wyjście drugiej warstwy:

$$a^2 = f^2(W^2 a^1 + b^2) = purelin\left(\begin{bmatrix} -0.45 & 0.33 \end{bmatrix} * \begin{bmatrix} 0.45 \\ 0.37 \end{bmatrix} + [0.39]\right) = [0.31]$$

Porównajmy odpowiedź sieci z wartością docelową.

$$e = t - a = (p-1)^2 - a^2 = (1-1)^2 - 0.31 = -0.31$$

Algorytm wstecznej propagacji koryguje parametry sieci tak, aby minimalizować powierzchnię funkcji błędu średniokwadratowego:

$$F(x) = E[e^2] = E[(t-a)^2]. \tag{2}$$

Ponieważ $F(x)$ jest funkcją zależności parametrów sieci: wag oraz biasów, ich korekcja w k-ej iteracji, m-ej warstwy powinna przebiegać zgodnie z zależnością:

$$w^m_{i,j}(k+1) = w^m_{i,j}(k) - \alpha * \frac{\delta F}{\delta w^m_{i,j}} \tag{3}$$

$$b^m_i(k+1) = b^m_i(k) - \alpha * \frac{\delta F}{\delta b^m_i} \tag{4}$$

gdzie α jest współczynnikiem uczenia.
Ponieważ $F(x)$ jest pośrednią funkcją wag oraz biasów w warstwie ukrytej, w celu ustalenia pochodnych przyjmijmy:

$$n_i^m = \sum w_{i,j}^m * a_j^{m-1} + b_i^m, \tag{5}$$

wejście do m-ej warstwy sieci, które jest bezpośrednio zależne od wag oraz biasów tej warstwy.
Tak więc pochodne możemy wyrazić w formie:

$$\frac{\delta F}{\delta w_{i,j}^m} = \frac{\delta F}{\delta n_i^m} \times \frac{\delta n_i^m}{\delta w_{i,j}^m} \tag{6}$$

$$\frac{\delta F}{\delta b_i^m} = \frac{\delta F}{\delta n_i^m} \times \frac{\delta n_i^m}{\delta b_i^m} \tag{7}$$

gdzie n jest bezpośrednio zależne od zmiennych w oraz b, więc z zależności 5 pochodne:

$$\frac{\delta n_i^m}{\delta w_{i,j}^m} = a_j^{m-1}, \ \frac{\delta n_i^m}{\delta b_i^m} = 1. \tag{8}$$

Tak więc zależności 6, 7 po przyjęciu, że:

$$s_i^m = \frac{\delta F}{\delta n_i^m}, \tag{9}$$

możemy przedstawić w postaci:

$$\frac{\delta F}{\delta w_{i,j}^m} = s_i^m * a_j^{m-1}, \tag{10}$$

$$\frac{\delta F}{\delta b_i^m} = s_i^m. \tag{11}$$

Ostatecznie, aktualizacja wag oraz biasów w formie macierzowej:

$$W^m(k+1) = W^m(k) - \alpha * s^m * (a^{m-1})^T, \tag{12}$$

$$b^m(k+1) = b^m(k) - \alpha * s^m. \tag{13}$$

Pozostało nam jeszcze do ustalenia wyrażenie s^m. Forma pochodnej w tym przypadku zależy od numeru bieżącej warstwy m. Rozpocznijmy od ostatniej warstwy, gdzie bezpośrednio ustalamy wartość błędu średniokwadratowego (wyr.2.):

$$s^M = \frac{\delta F}{\delta n^M} = -2 * F^M(n^M) * (t-a), \tag{14}$$

a następnie propagujemy do kolejnych warstw: $m = M-1, \ldots, 2, 1$ zgodnie z zależnością:

$$s^m = F^m(n^m) * (W^{m+1})^T * s^{m+1}. \tag{15}$$

Wróćmy do naszego przykładu, gdzie wyliczyliśmy odpowiedź sieci oraz porównaliśmy ją z wartością docelową (przyp. $e = t - a = (p-1)^2 - a^2 = (1-1)^2 - 0.31 = -0.31$). Możemy przejść do propagacji wstecz wartości błędu, w celu modyfikacji parametrów sieci. Wcześniej jednak ustalmy pochodne funkcji przejścia dla warstwy pierwszej (ukrytej): $f^1(n)$ oraz drugiej: $f^2(n)$. Dla pierwszej warstwy:

$$f^1(n) = \frac{d}{dn}(\frac{1}{1+e^{-n}}) = \frac{e^{-n}}{(1+e^{-n})^2} = (1 - \frac{1}{1+e^{-n}}) * \frac{1}{1+e^{-n}} = (1-a^1) * (a^1)$$

oraz drugiej:

$$f^2(n) = \frac{d}{dn}(n) = 1.$$

Wartość – wynikająca z błędu naszej aproksymacji – propagowana wstecz:

$$s^2 = -2 * F^2(n^2) * (t-a) = -2 * [f^2(n^2)] * (-0.31) = -2 * 1 * (-0.31) = 0.62.$$

Dla pierwszej warstwy z zależności 15.:

$$\begin{aligned} s^1 &= F^1(n^1) * (W^2)^T * s^2 = \begin{bmatrix} (1-a_1^1)(a_1^1) & 0 \\ 0 & (1-a_2^1)(a_2^1) \end{bmatrix} * \begin{bmatrix} -0.45 \\ 0.33 \end{bmatrix} * [0.62] \\ &= \begin{bmatrix} (1-0.45)(0.45) & 0 \\ 0 & (1-0.37)(0.37) \end{bmatrix} * \begin{bmatrix} -0.45 \\ 0.33 \end{bmatrix} * [0.62] \\ &= \begin{bmatrix} 0.247 & 0 \\ 0 & 0.233 \end{bmatrix} * \begin{bmatrix} -0.279 \\ 0.205 \end{bmatrix} = \begin{bmatrix} -0.069 \\ 0.048 \end{bmatrix} \end{aligned}$$

Teraz możemy dokonać modyfikacji parametrów sieci. Dla uproszczenia przyjmiemy współczynnik uczenia $\alpha = 0.1$. Z zależności 12., 13. mamy:

$$W^2(1) = W^2(0) - \alpha * s^2 * (a^1)^T = [-0.45\ 0.33] - 0.1 * [0.62] * [0.45\ 0.37] = [-0.4779\ 0.3071],$$

$$b^2(1) = b^2(0) - \alpha * s^2 = [0.39] - 0.1 * [0.62] = 0.328,$$

$$W^1(1) = W^1(0) - \alpha * s^1 * (a^0)^T = \begin{bmatrix} 0.09 \\ -0.37 \end{bmatrix} - 0.1 * \begin{bmatrix} -0.069 \\ 0.048 \end{bmatrix} * [1] = \begin{bmatrix} 0.0969 \\ -0.3748 \end{bmatrix}$$

$$b^1(1) = b^1(0) - \alpha * s^1 = \begin{bmatrix} -0.29 \\ -0.17 \end{bmatrix} - 0.1 * \begin{bmatrix} -0.069 \\ 0.048 \end{bmatrix} = \begin{bmatrix} -0.2831 \\ -0.1748 \end{bmatrix}.$$

W celu szerszego poeksperymentowania z obliczeniami algorytmu wstecznej propagacji błędu dla przedstawionej sieci, posłuż się programem demo Matlaba `nnd11bc` (Neural Network Design Demonstration Backpropagation Calculation). Zakończyliśmy pierwszą iterację algorytmu. W kolejnej iteracji podajemy na wejście sieci kolejną próbkę i kontynuujemy proces strojenia tak długo, dopóki nie uzyskamy satysfakcjonującej jakości działania.

1.3 Zadanie

Zgodnie z przedstawionym opisem należy wykonać implementację algorytmu wstecznej propagacji błędu w Matlabie. Sieć neuronowa powinna mieć topologię tak jak na rys. 3, z tymże o liczbie neuronów w pierwszej warstwie decyduje użytkownik. Następnie dla wybranego przykładu aproksymacji funkcji należy porównać czas obliczeń wykonany dla sieci własnej implementacji oraz implementacji Matlaba. Oczywiście analizę porównawczą należy wykonać dla sieci o identycznych założeniach (topologia, parametry, algorytm).

Literatura

[1] Mark H. Beale Martin T. Hagan, Howard B. Demuth. *Neural Network Design.* ISBN: 0-9717321-0-8.

[2] Osowski S. *Sieci neuronowe do przetwarzania informacji.* Oficyna Wydawnicza Politechniki Warszawskiej, 2006.