Zadanie Nr 1 – Wykorzystanie wielowarstwowego perceptronu

Inteligentna analiza danych

Jakub Sędziak, 203987 Filip Łuczak, 203929

12.04.2017

1. Cel zadania

Celem zadania było stworzenie sieci neuronowej która otrzymując na wejściu dane z Tabeli 1, poda te same dane na wyjściu.
Dla wartości wejściowych:

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Tabela 1

2. Wstęp Teoretyczny

Sieć neuronowa jest bardzo uproszczonym modelem mózgu. Składa się ona z dużej liczby (od kilkuset do kilkudziesięciu tysięcy) elementów przetwarzających informację. Elementy te nazywane są neuronami, chociaż w stosunku do rzeczywistych komórek nerwowych ich funkcje są bardzo uproszczone. Neurony są powiązane w sieć za pomocą połączeń o parametrach (tak zwanych wagach) modyfikowanych w trakcie tak zwanego procesu uczenia. Topologia połączeń oraz ich parametry stanowią program działania sieci, zaś sygnały pojawiające się na jej wyjściach w odpowiedzi na sygnały wejściowe są rozwiązaniami stawianych jej zadań. Większość współcześnie budowanych i wykorzystywanych sieci neuronowych ma budowę warstwową, przy czym ze względu na dostępność. w trakcie procesu uczenia wyróżnia się warstwy: wejściową, wyjściową oraz tak zwane warstwy ukryte. Każdy neuron posiada dowolna liczbę wejść oraz jedno wyjście. Neurony pogrupowane są w warstwy, gdzie każde dwie sąsiednie warstwy połączone są ze sobą. Na wejście neuronu trafiają wyjścia wszystkich neuronów z warstwy poprzedniej.

2.1Wsteczna Propagacja

Metoda wstecznej propagacji błędów polega na doborze odpowiednich wag i nanoszeniu poprawek, poprzez wyliczanie błędu każdego neuronu oraz jego delty, zaczynając od neuronów wyjściowych. Wielkość błędu w każdej warstwie jest zależny od błędu w warstwie następującej. Wartość błędu wykorzystywana jest do obliczenia delty, a ta z kolei do zmiany wag wejść danej warstwy. W przypadku sieci typu perceptron szczególnie przydatną ze względu na fakt potencjalnej możliwości bardzo efektywnej implementacji numerycznej, techniką optymalizacyjną jest lokalna metoda najszybszego spadku gradientu. Polega ona na stopniowym, iteracyjnym poprawianiu położenia punktu przybliżenia w przestrzeni wag sieci przez dodawanie w każdym jej kroku do aktualnego przybliżenia rozwiązania fragmentu ujemnego gradientu funkcji błędu w tym punkcie. Zakładając dostatecznie małą wielkość kroku optymalizacyjnego postępowanie takie gwarantuje zmniejszanie się wartości funkcji błędu w każdej iteracji omawianej metody. Mimo, że procedura ta nie jest w ogólnym przypadku zbieżna do najlepszego z możliwych przybliżeń rozwiązania problemu, chociażby ze względu na potencjalne niebezpieczeństwo jej utknięcia w jednym z minimów lokalnych funkcji błędu sieci, to w praktyce wykorzystanie omawianej procedury prowadzi często do znalezienia punktu w przestrzeni wag sieci przybliżającego rozwiązanie idealne z dużą, satysfakcjonującą jej użytkownika dokładnością.

2.2Wzory

Obliczenie błędu między wartością oczekiwaną a wartością obliczoną przez sieć neuronową:

$$\underset{w\epsilon W}{argmin}\left\{E: W \to R;\ E(w) \triangleq \frac{1}{2}\sum_{\mu=1}^{N}\sum_{i=1}^{m_n}\left(z_{i\mu} - v_{i\mu}^{(n)}\right)^2\right\} \tag{1}$$

Zakładając istnienie rozwiązania problemu (1), z podanej wyżej definicji funkcji błędu E sieci wynika natychmiast, że jej wartością minimalną jest zero, oraz wartość ta osiągana jest w punktach w ∈ W przestrzeni wag sieci, w których odpowiedzi na wszystkich wyjściach sieci są identyczne z zadanymi odpowiedziami spodziewanymi dla każdego ze wzorców uczących, stanowiących elementy zbioru uczącego Ω . Problem (1) sprowadza się zatem do wskazania dowolnego wektora w ∈ W wag rozważanej sieci, dla którego spełniony jest warunek: E (w) = 0. W ogólnym przypadku, nawet jeśli nie istnieje dokładne rozwiązanie problemu (1) to zawsze można próbować znaleźć jego przybliżone rozwiązanie stosując powszechnie znane numeryczne metody optymalizacji nieliniowej. Jedną z takich metod jest metoda najszybszego spadku gradientu. W tym celu należy wyznaczyć postać ujemnego gradientu.

Aby wyznaczyć postać ujemnego gradientu – ∇E funkcji błędu E sieci w dowolnym punkcie w ∈ W przestrzeni wag należy najpierw wyznaczyć deltę w sposób następujący:

$$\delta_{i\mu}^{(k)} = \begin{cases} f'\left(s_{i\mu}^{(k)}\right) * \left(z_{i\mu} - v_{i\mu}^{(k)}\right) dla\ k = n \\ f'\left(s_{i\mu}^{(k)}\right) * \sum_{j=1}^{m_{k+1}} \left(w_{j\,i}^{(k+1)} - \delta_{j\mu}^{(k+1)}\right) dla\ k = 1, \dots, n-1 \end{cases} \tag{2}$$

gdzie: $\delta_{i\mu}^{(k)}$ określać będziemy mianem sygnału zwrotnego dla i - tego neuronu k - tej warstwy sieci przy μ - tym wzorcu uczącym.

$z_{i\mu}$ – wartość oczekiwana i - tego neuronu przy μ - tym wzorcu uczącym.

$v_{i\mu}^{(k)}$ – wyjście i - tego neuronu k - tej warstwy sieci przy μ - tym wzorcu uczącym.

$w_{j\,i}^{(k+1)}$ – j - ta waga i - tego neuronu k - tej warstwy sieci

f’ – Jest to pochodna cząstkowa funkcji aktywacji.

Korzystamy z dwóch funkcji aktywacji :

Liniowej

$$f(x) = x \tag{3a}$$

$$f'(x) = 1 \tag{3b}$$

Sigmoidalnej

$$f(x) = \frac{1}{1+e^{-Bx}} \tag{4a}$$

$$f'(x) = \frac{B*e^{-Bx}}{(1+e^{-Bx})^2} \tag{4b}$$

Parametr B ma wpływ na kształt funkcji aktywacji i w naszym przypadku wynosi 1. x – jest to suma wejść oraz wag w neuronie.

Wzór poprawki wektora wagowego pojedynczego kroku nauki sieci metodą on-line ma postać:

$$w_{ji}^{(k)}(t+1) = w_{ji}^{(k)}(t) + \eta\delta_{i\mu_0}^{(k)}(t)v_{j\mu_0}^{(k-1)}(t);\ \ \mu_0 \epsilon \{1, \dots, N\} \tag{5a}$$

$$E_{\mu_0}(w) \triangleq \frac{1}{2}\sum_{i=1}^{m_n}\left(z_{i\mu_o} - v_{i\mu_0}^{(k)}\right)^2 \tag{5b}$$

gdzie: E_{μ_0} jest pojedynczym składnikiem funkcji błędu (1) dla ustalonego wzorca uczącego μ_o, zaś krok iteracji metody t ∈ N może być utożsamiany z prezentacją uczonej sieci pojedynczego wzorca treningowego. η - jest współczynnikiem nauki

2.3 Algorytm

Poniżej przedstawiono algorytm wstecznej propagacji błędu oparty o metodę on-line:

1) Wybierz niewielką wartość kroku nauki η (np. $\eta = 0.7$), zaś początkowe wartości wszystkich wag sieci wybierz jako małe liczby losowe (np. z przedziału [-1 , 1]) .

2) Wybierz losowo wzorzec uczący ($[v_{1\mu_0}^{(0)} \dots v_{m_0\mu_0}^{(0)}]$, $[z_{1\mu_0} \dots z_{m_0\mu_0}]$) $\in \Omega$; $\mu_0 \in \{1, \dots, N\}$ ze zbioru treningowego Ω i przepuść sygnały wejściowe $v_{1\mu_0}^{(0)} \dots v_{m_0\mu_0}^{(0)}$ przez sieć w przód wyznaczając i zapamiętując wyjścia $v_{1\mu_0}^{(k)}$ i sumy ważone $s_{i\mu_0}^{(k)}$ dla wszystkich neuronów sieci.

3) Oblicz sygnały zwrotne $s_{i\mu_0}^{(n)}$ wszystkich neuronów warstwy wyjściowej sieci korzystając ze wzoru:

$$\delta_{i\mu}^{(k)} = f'\left(s_{i\mu}^{(k)}\right) * \left(z_{i\mu} - v_{i\mu}^{(k)}\right) dla\ k = n$$

4) Oblicz sygnały zwrotne $\delta_{i\mu}^{(k)}$ wszystkich neuronów warstw poprzednich sieci propagując te sygnały kolejno wstecz sieci poczynając od warstwy n –1 aż do warstwy 1 za pomocą wzoru:

$$\delta_{i\mu}^{(k)} = f'\left(s_{i\mu}^{(k)}\right) * \sum_{j=1}^{m_{k+1}} \left(w_{j\,i}^{(k+1)} - \delta_{j\mu}^{(k+1)}\right) dla\ k = 1, \dots, n-1$$

5) Korzystając z wyznaczonych i zapamiętanych w punktach 2) - 4) wielkości wyjść $v_{1\mu_0}^{(k)}$ i sygnałów zwrotnych $\delta_{i\mu}^{(k)}$ neuronów sieci dokonaj zmiany każdej z wag uczonej sieci według wzoru:

$$w_{ji}^{(k)}(t+1) = w_{ji}^{(k)}(t) + \eta\delta_{i\mu_0}^{(k)}(t)v_{j\mu_0}^{(k-1)}(t);\ \ \mu_0 \in \{1, \dots, N\}$$

6) Jeśli nie wyczerpano wszystkich wzorców uczących ze zbioru Ω to wybierz losowo kolejny wzorzec, nie podawany jeszcze na wejście sieci, i przejdź do punktu 2), w przeciwnym razie idź do punktu 7).

7) Czy sieć odtwarza z założoną dokładnością każdy ze wzorców treningowych? Jeśli tak to koniec. W przeciwnym razie rozpocznij kolejną epokę nauczania sieci przechodząc do punktu 2)

2.4 Domyślne parametry sieci:

Współczynnik nauki :	0.3
Liczba neuronów w warstwie wejściowej :	4
Liczba neuronów w warstwie ukrytej :	3
Liczba neuronów w warstwie wyjściowej :	4
Ilość iteracji/epok :	10000
Dokładność / dopuszczalny błąd :	0.01
Zastosowany algorytm uczenia się :	Wsteczna propagacja błędu wykorzystująca metodę najszybszego spadku gradientu, metoda on-line
Momentum :	brak
Bias :	brak

Tabela 2

Funkcje aktywacji:

Neurony wejściowe :	funkcja liniowa
Neurony w warstwie ukrytej oraz wyjściowej :	funkcja sigmoidalna

Tabela 3

Parametry te są domyślne i nie zmieniają się, chyba ze w eksperymencie jest napisane inaczej.

3. Eksperyment i wyniki

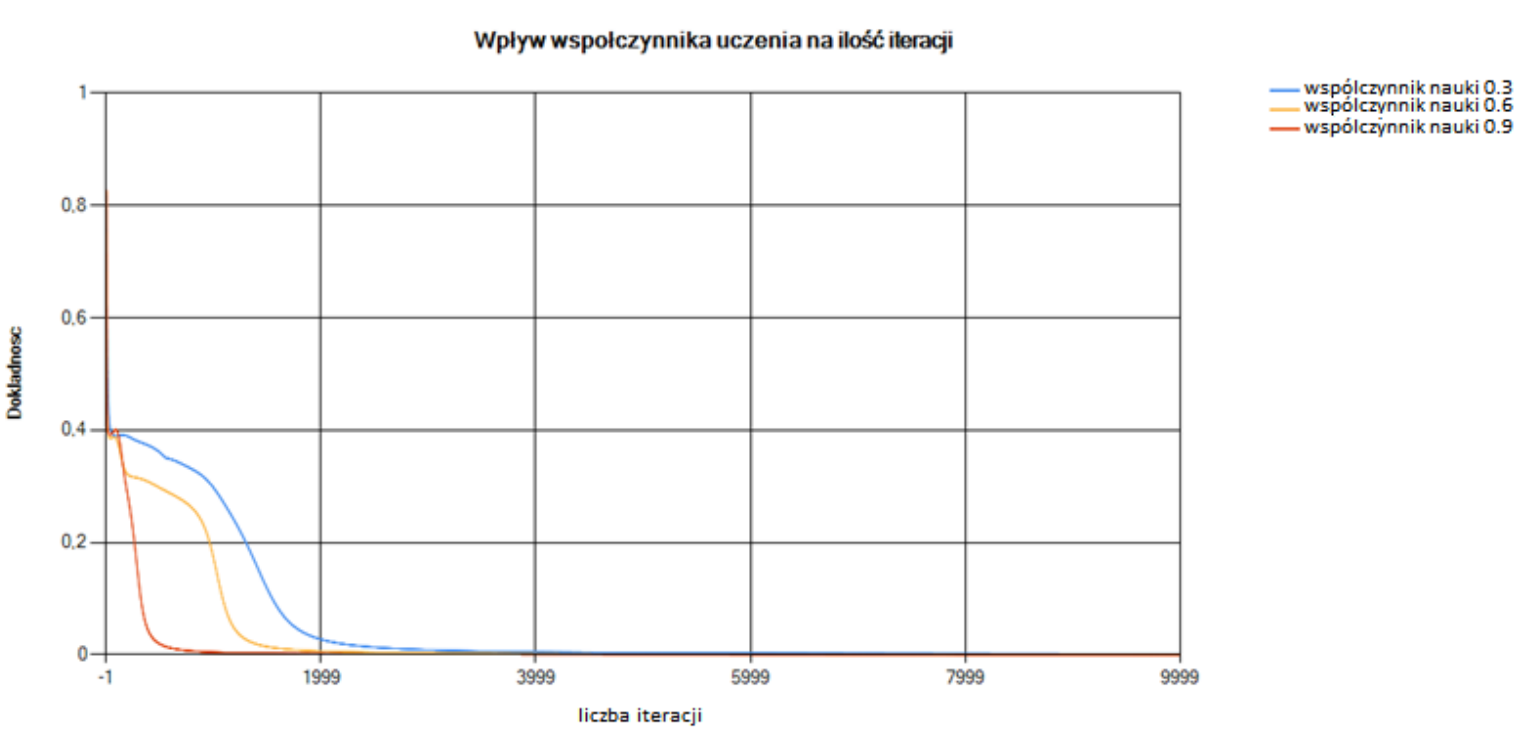
3.1 Eksperyment nr 1

3.1.1 Założenia

Wpływ współczynnika prędkości nauki na liczbę epok potrzebnych do nauczenia sieci.

Przyjęliśmy trzy różne wartości współczynnika nauki wynoszące kolejno 0.3, 0.6 oraz 0.9.

3.1.2 Przebieg



3.1.3 Rezultat

Wartość współczynnika nauki	Epoka w której osiągnięto błąd < 0.01
0.3	2799
0.6	1672
0.9	651

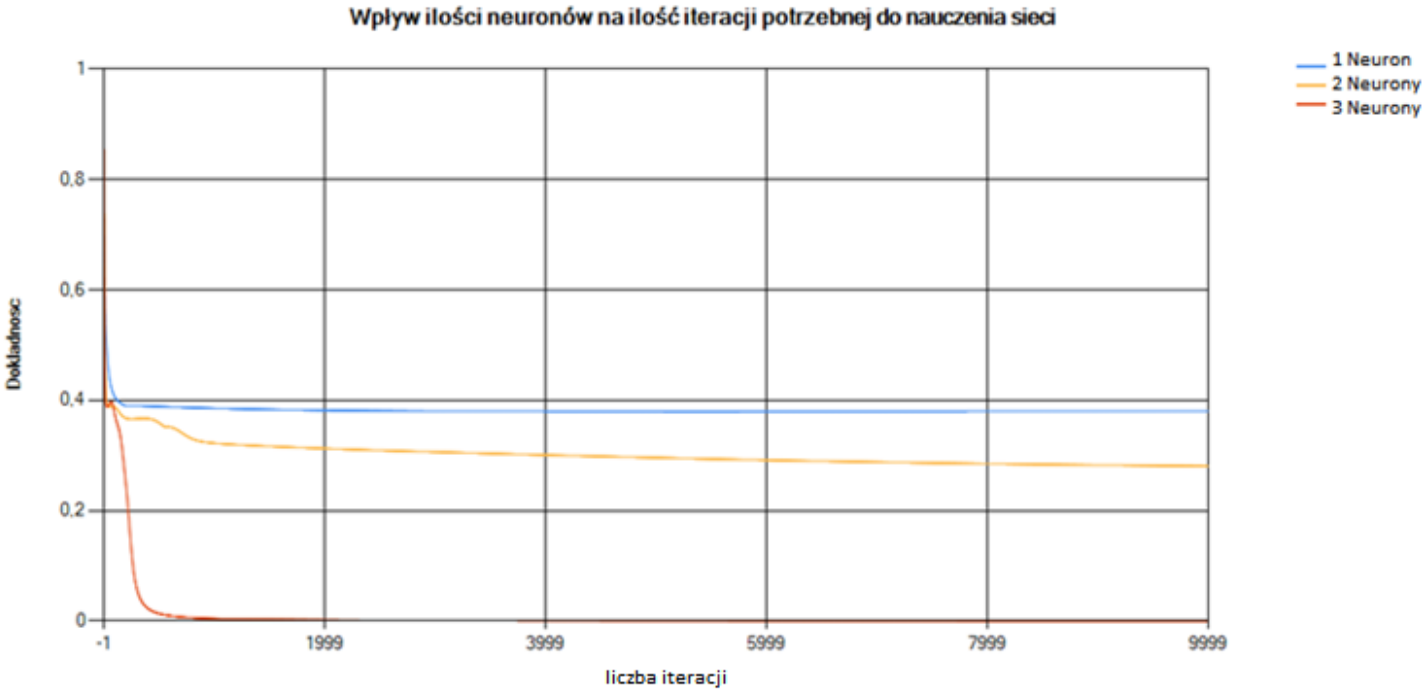
Tabela 4

3.2 Eksperyment nr 2

3.2.1 Założenia

Wpływ ilości Neuronów, od 1 do 3, na prędkość uczenia się sieci bez biasu.

3.2.2 Przebieg



3.2.3 Rezultat

Ilość neuronów w warstwie ukrytej	Osiągnięto dokładność w epoce nr:
1	Nie osiągnięto żądanej dokładności
2	Nie osiągnięto żądanej dokładności
3	575

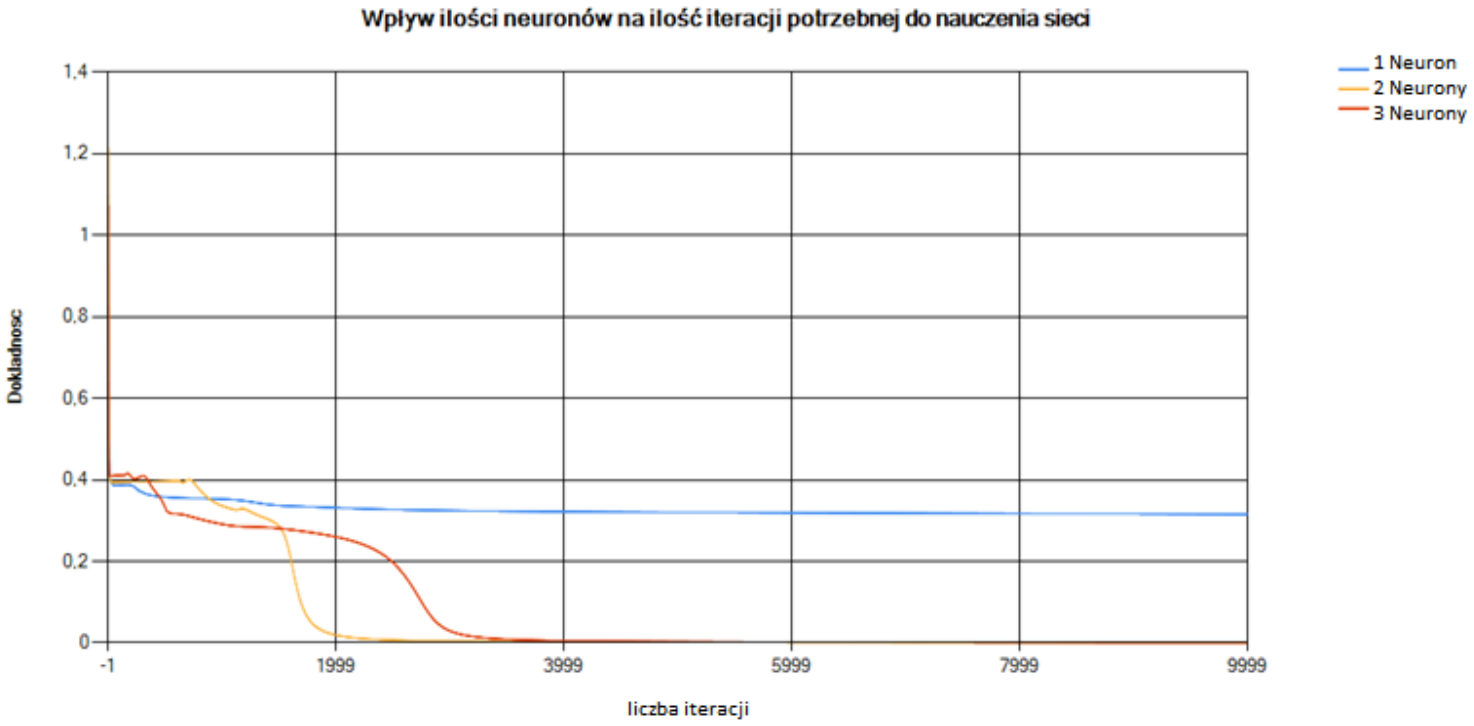
Tabela 5

3.3 Eksperyment nr 3

3.3.1 Założenia

Wpływ biasu na uczenie się sieci neuronowej, posiadającej 1, 2 lub 3 neurony.

3.3.2 Przebieg



3.3.3 Rezultat

Ilość neuronów w warstwie ukrytej	Osiągnięto dokładność w epoce nr:
1	Nie osiągnięto żądanej dokładności
2	2248
3	3406

Tabela 6

3.4 Eksperyment nr 4

3.4.1 Założenia

Interpretacja wyjść z warstwy ukrytej. Wykorzystując 2 neurony w warstwie ukrytej.

3.4.2 Przebieg

Bez Biasu (sieć nie osiągnęła żądanego przybliżenia):

Dane:	Wyjście 1	Wyjście 2
1000	0,388	0,417
0100	0,995	0,418
0010	0,993	0,350
0001	0,259	0,335

Tabela 7

Z Biasem:

Dane:	Wyjście 1	Wyjście 2
1000	0,950	0,754
0100	0,093	0,857
0010	0,029	0,195
0001	0,689	0,149

Tabela 8

3.4.3 Rezultat

Z danych w tabeli 7 oraz tabeli 8 możemy łatwo wywnioskować, że dla biasu i 2 neuronów w warstwie ukrytej, wszystkie wejścia zostały prawidłowo rozpoznane . Wyjścia są bliskie 1 i 0 oraz występują na przemian lub powtarzają się. Takiej zależności nie ma dla testu sieci bez biasu. Dlatego nie wszystkie wzorce zostaną odtworzone na wyjściu.

4. Wnioski

Na podstawie przeprowadzonych eksperymentów doszliśmy do następujących wniosków:

- Im wyższy współczynnik nauki tym sieć uczy się szybciej, lecz istnieje ryzyko że sieć „przeskoczy” poszukiwaną wagę i będzie wokół niej oscylować.
- Większa liczba neuronów poprawia rezultat oraz przyspiesza uczenie się sieci.
- Zbyt małe współczynniki mogą spowodować że sieć utknie w minimum lokalnym, i nigdy nie osiągnie minimum globalnego.
- Stosowanie neuronu obciążającego (biasu) wspomaga działanie sieci, i umożliwia rozwiązanie części problemów.
- Gdy błąd sieci przestaje maleć, można przerwać dalsze poszukiwanie minimum i uczenie sieci.

Bilbiografia

Ryszard Tadeusiewcz "Sieci neuronowe", Kraków 1992
Józef Korbicz, Andrzej Obuchowicz, Dariusz Uciński "Sztuczne sieci neuronowe: podstawy i zastosowania”, Warszawa 1994