

Ryszard Tadeusiewicz

Katedra Automatyki  
Akademia Górnictwo-Hutnicza  
Kraków

## Sieci neuronowe\*

### Neural networks

*Abstract:* Neural network is a new tool for parallel data processing and for modelling of real brain. The recent resurgence of interest in neural networks has its roots in the recognition that the brain performs computations in a different manner than a conventional digital computer. The paper presents fundamental concepts of neural network, its learning rules and applications.

#### 1. Wprowadzenie

Sieci neuronowe są nowymi, pracującymi współbieżnie systemami przetwarzania informacji. Mogą one także być rozpatrywane jako modele wybranych fragmentów i wybranych funkcji systemu nerwowego. Jedno i drugie powoduje, że sieci te budzą powszechnie zainteresowanie, gdyż przez jednych traktowane są jako zapowiedź komputerów XXI wieku, a dla innych mają walor klucza do zrozumienia tajników ich własnego intelektu. Pozostawiając chwilowo na uboczu kontrowersyjną kwestię przydatności sieci neuronowych jako modeli ludzkiego mózgu, można bez obawy stwierdzić, że są to układy przetwarzające informacje, mające w stosunku do typowych systemów obliczeniowych dwie zasadnicze zalety.

Po pierwsze, obliczenia są w sieciach neuronowych wykonywane równolegle; tysiące sztucznych neuronów składających się na sieć, wykonują przypadające im zadania obliczeniowe równocześnie, w związku z czym szybkość pracy sieci neuronowych może miliony razy przewyższać szybkość aktualnie używanych komputerów.

---

\*Referat wygłoszony na XXXII Zjeździe Fizyków Polskich w Krakowie we wrześniu 1993 r.

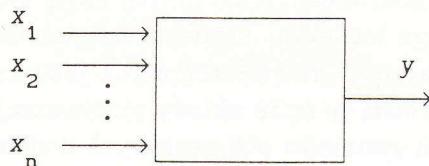
Po drugie, sieci nie trzeba programować. Liczne znane obecnie metody uczenia i samouczenia sieci pozwalają uzyskać ich celowe i skuteczne działanie nawet w sytuacji, kiedy twórca sieci nie zna algorytmu, według którego można rozwiązać postawione zadanie.

Wymienione zalety spowodowały w ostatnich latach ogromny wzrost zainteresowania tą problematyką. Na świecie powstały periodyki naukowe, poświęcone tylko sieciom neuronowym, zwoływano dziesiątki specjalistycznych sympozjów i konferencji, wydano setki książek, uruchomiono tysiące projektów badawczych. Wyczerpujące zreferowanie tak obszernej dziedziny w jednym krótkim artykule jest oczywiście niemożliwe, dlatego zainteresowanych Czytelników zachęcam do siegnięcia po książkę [1].<sup>1</sup>

## 2. Czym jest i jak działa sieć neuronowa?

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, by nie powiedzieć – sprymitywizowane. Neurony są powiązane w sieć za pomocą połączeń o parametrach (tzw. wagach) modyfikowanych w trakcie procesu uczenia. Topologia połączeń oraz ich parametry stanowią program działania sieci, zaś sygnały pojawiające się na jej wyjściach w odpowiedzi na określone sygnały wejściowe są rozwiązaniami stawianych jej zadań.

Elementy, z których buduje się sieci, charakteryzują się występowaniem wielu wejść i jednego wyjścia (rys. 1).



Rys. 1. Ogólna budowa neuronu

Sygnały wejściowe  $x_i$  ( $i = 1, 2, \dots, n$ ) oraz sygnał wyjściowy  $y$  mogą przyjmować wartości z pewnego ograniczonego przedziału; z dokładnością do prostej

<sup>1</sup> Patrz także artykuły: B. Macukow „Neurokomputery”, *Postępy Fizyki* **41**, 265 (1990), T. Szoplik „Połączenia optyczne w komputerach” *ibid.* **42**, 165 (1991), K. Chałasińska-Macukow „Optyczne pамięci skojarzeniowe” *ibid.* **42**, 605 (1991) (przyp. Red.).

funkcji skalującej można przyjąć, że

$$x_i \in [-1, 1].$$

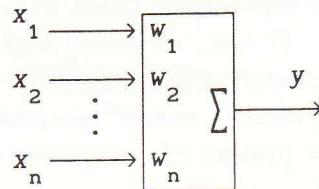
Zależność

$$y = f(x_1, x_2, \dots, x_n)$$

w najprostszym przypadku może być rozważana jako liniowa:

$$y = \sum_{i=1}^n w_i x_i,$$

przy czym współczynniki  $w_i$ , nazywane wagami synaptycznymi, mogą podlegać modyfikacjom w trakcie procesu uczenia (rys. 2).



Rys. 2. Neuron liniowy

Wprowadzając w dalszych rozważaniach wektory sygnałów wejściowych

$$\mathbf{X} = \langle x_1, x_2, \dots, x_n \rangle^T$$

i współczynników wagowych

$$\mathbf{W} = \langle w_1, w_2, \dots, w_n \rangle^T,$$

funkcję liniowego neuronu można opisać wzorem

$$y = \mathbf{W}^T \mathbf{X}.$$

Z ogólnie znanych właściwości iloczynu skalarnego wynika, że sygnał wyjściowy neuronu  $y$  będzie tym większy, im bardziej położenie wektora wejściowego  $\mathbf{X}$  w przestrzeni  $\mathbf{X}$  przypominać będzie położenie wektora wag  $\mathbf{W}$  w przestrzeni  $\mathbf{W}$ .

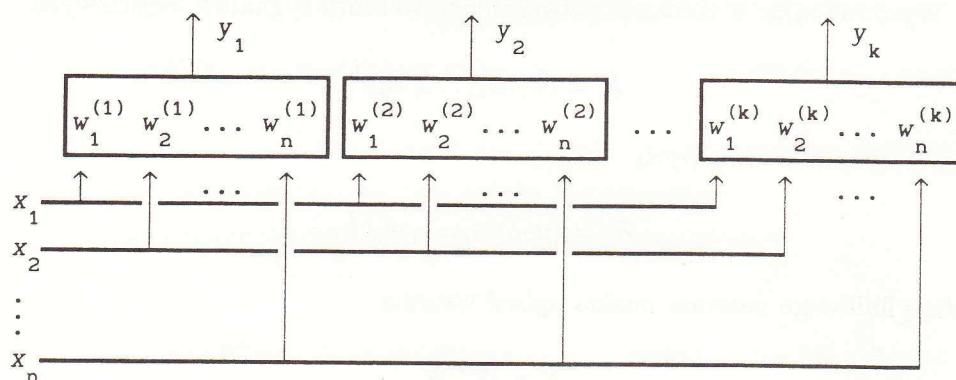
W ten sposób można powiedzieć, że neuron rozpoznaje<sup>2</sup> sygnały wejściowe, wyróżniając te, które są podobne do jego wektora wag.

Rozważmy teraz warstwę neuronów, z których każdy ma ten sam zestaw sygnałów wejściowych  $\mathbf{X} = \langle x_1, x_2, \dots, x_n \rangle^T$ , natomiast każdy ma swój własny wektor wag. Ponumerujmy neurony w warstwie i oznaczmy jako  $\mathbf{W}^{(m)} = \langle w_1^{(m)}, w_2^{(m)}, \dots, w_n^{(m)} \rangle^T$  wektor wag  $m$ -tego neuronu ( $m = 1, 2, \dots, k$ ). Wówczas oczywiście sygnał wyjściowy  $m$ -tego neuronu można wyznaczyć ze wzoru

$$y^{(m)} = \sum_{i=1}^n w_i^{(m)} x_i.$$

Omawiana warstwa stanowi najprostszy przykład sieci neuronowej.

Działanie tej sieci polega na tym, że pojawienie się określonego wektora wejściowego  $\mathbf{X}$  powoduje powstanie sygnałów wyjściowych  $y_m$  na wszystkich neuronach wchodzących w skład rozważanej warstwy. Oczekujemy przy tym maksymalnego sygnału wyjściowego  $y_m$  na tym neuronie, którego wektor wag  $\mathbf{W}^{(m)}$  najbardziej przypomina  $\mathbf{X}$ . Sieć tego typu może więc rozpoznawać  $k$  różnych klas obiektów, gdyż każdy neuron zapamiętuje jeden wzorcowy obiekt, na którego pojawienie się jest „uczulony”. O tym, do której klasy należy zaliczyć aktualnie pokazany obiekt, decyduje numer wyjścia, na którym pojawia się sygnał  $y_m$  o maksymalnej wartości. Oczywiście „wzorce” poszczególnych klas zawarte są w poszczególnych neuronach w postaci ich wektorów wag  $\mathbf{W}^{(m)}$ . Schemat takiej sieci przedstawia rys. 3.



Rys. 3. Struktura sieci liniowej

<sup>2</sup> Składowe sygnału wejściowego  $\mathbf{X}$  można rozważać jako cechy pewnych obiektów, a sygnał wyjściowy  $y$  może stanowić miarę podobieństwa tych obiektów do pewnej wyróżnionej klasy. Interpretacja tego typu nawiązuje do często stosowanego modelu *rozpoznawania obrazów* [2].

Stosując konsekwentnie notację wektorową można sygnały wyjściowe z rozważanej warstwy neuronów zebrać w formie wektora

$$\mathbf{Y} = \langle y_1, y_2, \dots, y_k \rangle^T.$$

Wektor ten można wyznaczyć mnożąc wektor wejściowy  $\mathbf{X}$  przez macierz  $\mathbf{W}_k$  o wymiarach  $[k \times n]$ , utworzoną w taki sposób, że jej kolejnymi wierszami są (transponowane) wektory  $\mathbf{W}^{(m)}$  (dla  $m = 1, 2, \dots, k$ ). Macierz  $\mathbf{W}_k$  ma więc następującą budowę:

$$\mathbf{W}_k = \begin{pmatrix} w_1^{(1)} & w_2^{(1)} & \dots & w_n^{(1)} \\ w_1^{(2)} & w_2^{(2)} & \dots & w_n^{(2)} \\ \vdots & \vdots & & \vdots \\ w_1^{(k)} & w_2^{(k)} & \dots & w_n^{(k)} \end{pmatrix}.$$

Wykorzystując macierz  $\mathbf{W}_k$  można zapisać funkcje realizowane przez całą sieć w formie jednego zwarteego wzoru:

$$\mathbf{Y} = \mathbf{W}_k \mathbf{X}.$$

Obiektem podlegającym uczeniu w takiej sieci jest *macierz*  $\mathbf{W}_k$ , której wartości trzeba tak dobrać, by sieć realizowała założoną formę odwzorowania sygnałów wejściowych  $\mathbf{X}$  na sygnały wyjściowe  $\mathbf{Y}$ . Podczas uczenia wykorzystuje się tak zwany ciąg uczący

$$U = \langle \langle \mathbf{X}^{(1)}, \mathbf{Z}^{(1)} \rangle, \langle \mathbf{X}^{(2)}, \mathbf{Z}^{(2)} \rangle, \dots, \langle \mathbf{X}^{(N)}, \mathbf{Z}^{(N)} \rangle \rangle,$$

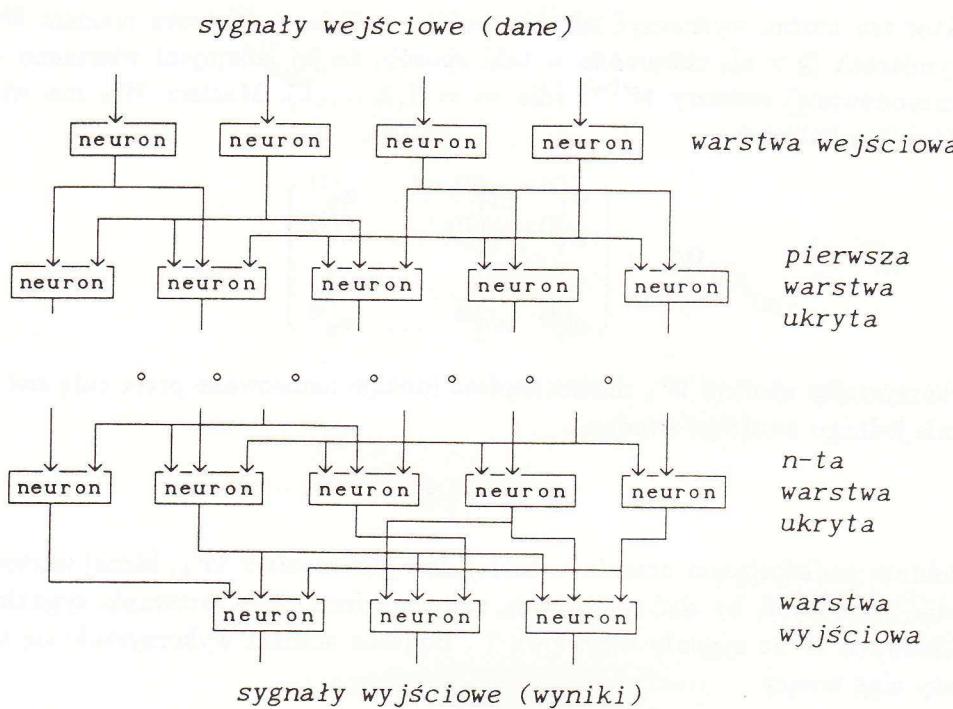
gdzie  $\mathbf{Z}^{(j)}$  są  $k$ -elementowymi wektorami oznaczającymi wymagane zestawy odpowiedzi sieci na wymuszenia dane odpowiednimi wektorami  $\mathbf{X}^{(j)}$ . Uczenie sieci opisuje formuła:

$$\mathbf{W}_k^{(j+1)} = \mathbf{W}_k^{(j)} + \eta (\mathbf{Z}^{(j)} - \mathbf{Y}^{(j)}) (\mathbf{X}^{(j)})^T.$$

Wektor  $\mathbf{X}$  jest  $n$  elementowy i po transpozycji tworzy wiersz o  $n$  kolumnach. Wektory  $\mathbf{Z}$  oraz  $\mathbf{Y}$  są  $k$  elementowe, więc ich różnica tworzy kolumnę o  $k$  wierszach. Przemnożenie tych elementów przez siebie tworzy macierz poprawek  $\Delta \mathbf{W}_k$  o rozmiarach  $[k \times n]$  – dokładnie zgodną z wymiarami macierzy  $\mathbf{W}_k$ , co umożliwia poprawne dodawanie tych macierzy.

Opisana procedura pozwala na uczenie sieci dowolnych liniowych odwzorowań  $\mathbf{X} \rightarrow \mathbf{Y}$ . Znacznie szersze możliwości wiążą się jednak ze stosowaniem sieci

elementów nieliniowych. Sieci takich elementów mają budowę wielowarstwową,<sup>3</sup> przy czym ze względu na dostępność w trakcie procesu uczenia wyróżnia się warstwy: wejściową, wyjściową oraz tak zwane warstwy ukryte (rys. 4).



Rys. 4. Wielowarstwowa sieć elementów nieliniowych

Nieliniowy element przyjmowany w sieciach neuronowych może być opisany równaniem

$$y = \varphi(e),$$

gdzie  $\varphi(\cdot)$  jest wybraną funkcją nieliniową a sygnał  $e$  odpowiada łącznemu pobudzeniu neuronu

$$e = \sum_{i=1}^n w_i x_i + w_0.$$

<sup>3</sup> Sieci liniowe nie mogą mieć struktury wielowarstwowej, ponieważ łatwo stwierdzić, że liniowa struktura wielowarstwowa ma dokładnie te same możliwości przetwarzania informacji, co sieć jednowarstwowa i wprowadzanie dodatkowych warstw sieci jest bezcelowe.

Aby uprościć występujące dalej wzory przyjmiemy, że obok sygnałów  $\langle x_1, x_2, \dots, x_n \rangle$  składających się na wektor  $\mathbf{X}$ , występować będzie składnik  $x_0 \equiv 1$ . Ten formalny zabieg pozwoli zapisać sygnał  $e$  prostym wzorem

$$e = \sum_{i=0}^n w_i x_i.$$

Wzór opisujący uczenie (zmianę wartości składowych wektora wag takiego neuronu) podczas przedstawiania próbek z ciągu uczącego

$$U = \langle \langle \mathbf{X}^{(1)}, z^{(1)} \rangle, \langle \mathbf{X}^{(2)}, z^{(2)} \rangle, \dots, \langle \mathbf{X}^{(N)}, z^{(N)} \rangle \rangle,$$

wygodnie będzie wyprowadzić początkowo dla jednego neuronu. Formuły uczenia szukać będziemy opierając się na regule minimalizacji funkcjonału błędu średnio-kwadratowego:

$$Q = \frac{1}{2} \sum_{j=1}^N \left( z^{(j)} - y^{(j)} \right)^2,$$

gdzie oczywiście

$$y^{(j)} = \varphi \left( \sum_{i=0}^n w_i^{(j)} x_i^{(j)} \right).$$

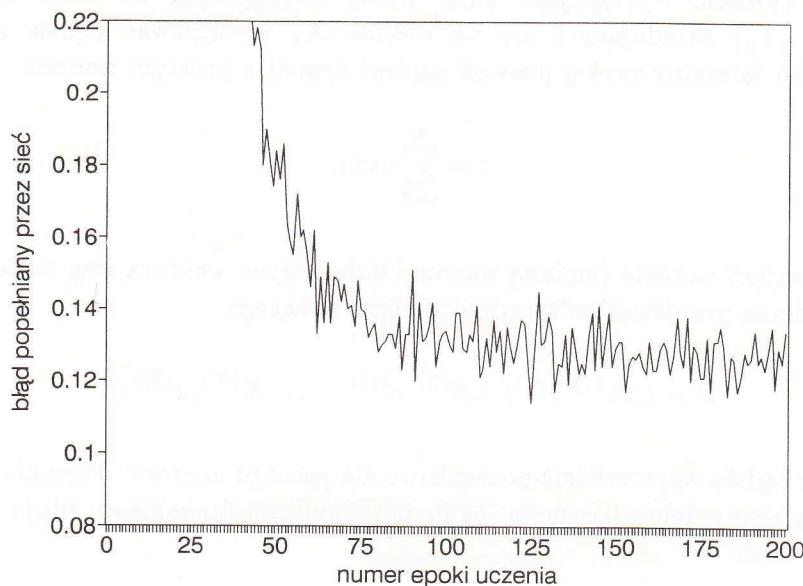
Rozłożmy funkcjonał błędu na elementy składowe związane z poszczególnymi krokami procesu uczenia

$$Q = \sum_{j=1}^N Q^{(j)},$$

gdzie

$$Q^{(j)} = \frac{1}{2} \left( z^{(j)} - y^{(j)} \right)^2.$$

Proces uczenia prowadzi do minimalizacji błędu  $Q$  popełnianego przez sieć, jak to przedstawiono przykładowo na rys. 5.



Rys. 5. Minimalizacja błędu w trakcie procesu uczenia

Dla osiągnięcia tego efektu możemy postępować zgodnie z gradientową strategią procesu uczenia. Strategię tę możemy zapisać w formie algorytmu zmian współczynników wag w stopniu proporcjonalnym do odpowiednich składowych  $\partial Q / \partial w$ , ale oczywiście w przeciwnym kierunku

$$w_i^{(j+1)} - w_i^{(j)} = \Delta w_i^{(j)} = -\eta \frac{\partial Q^{(j)}}{\partial w_i}.$$

Ponieważ zależność sygnału  $y$  od współczynników wag  $w_i$  ma uwikłany charakter, uwzględniający nieliniową funkcję  $\varphi(e)$ , pochodną cząstkową wylicza się ze wzoru

$$\frac{\partial Q^{(j)}}{\partial w_i} = \frac{\partial Q^{(j)}}{\partial y^{(j)}} \frac{\partial y^{(j)}}{\partial w_i} = \frac{\partial Q^{(j)}}{\partial y^{(j)}} \frac{dy^{(j)}}{de^{(j)}} \frac{\partial e^{(j)}}{\partial w_i}.$$

Pierwszą pochodną występującą w omawianym wzorze można obliczyć bardzo łatwo ze wzoru definiującego  $Q^{(j)}$

$$\frac{\partial Q^{(j)}}{\partial y^{(j)}} = -(z^{(j)} - y^{(j)}) = -\delta^{(j)},$$

gdzie  $\delta^{(j)}$  uznać można za wielkość błędu popełnianego przez sieć w  $j$ -tym kroku procesu uczenia.

Ostatni z rozważanych czynników też łatwo wyznaczyć

$$\frac{\partial e^{(j)}}{\partial w_i} = x_i^{(j)},$$

ponieważ

$$e^{(j)} = \sum_{i=0}^n w_i^{(j)} x_i^{(j)}.$$

Jedyny problem może powstać z czynnikiem  $dy^{(j)}/de^{(j)}$ , ponieważ oczywiście

$$\frac{dy^{(j)}}{de^{(j)}} = \frac{d\varphi(e)}{de^{(j)}}$$

a funkcja  $\varphi(e)$  nie zawsze jest różniczkowalna. Dlatego tak chętnie (i trochę bezkrytycznie) przyjmowana jest w rozważaniach nad sieciami neuronowymi funkcja logistyczna

$$y = \varphi(e) = \frac{1}{1 + \exp(-\beta e)}$$

ponieważ zaletą tej funkcji jest prosta i łatwa do obliczenia wartość jej pochodnej

$$\frac{d\varphi(e)}{de^{(j)}} = y^{(j)}(1 - y^{(j)}).$$

Ostateczny wzór, na podstawie którego uczy się sieci złożone z nieliniowych elementów, może być zapisany w postaci

$$\Delta w_i^{(j)} = \eta \delta^{(j)} \frac{d\varphi(e)}{de^{(j)}} x_i^{(j)}.$$

Wzór ten dla funkcji logistycznej zapisany może być w prostszej postaci

$$\Delta w_i^{(j)} = \eta (z^{(j)} - y^{(j)}) (1 - y^{(j)}) x_i^{(j)} y^{(j)}$$

bezpośrednio użytecznej przy konstrukcji elementów naśladowujących sieci neuronowe lub algorytmów symulujących te sieci.

Opisany wyżej algorytm uczenia jest możliwy do bezpośredniego zastosowania jedynie w przypadku sieci jednowarstwowej. Dla sieci wielowarstwowych podany wyżej wzór nie daje się zastosować, ponieważ dla wewnętrznych warstw sieci nie ma możliwości bezpośredniego określenia oczekiwanych (wymaganych)

wartości sygnałów wyjściowych  $z^{(j)}$ , a tym samym nie ma możliwości określenia błędu  $\delta^{(j)}$ . Istotnie, rozważając ciąg uczący w postaci

$$U = \langle\langle \mathbf{X}^{(1)}, \mathbf{Z}^{(1)} \rangle, \langle \mathbf{X}^{(2)}, \mathbf{Z}^{(2)} \rangle, \dots, \langle \mathbf{X}^{(M)}, \mathbf{Z}^{(M)} \rangle \rangle$$

mamy do dyspozycji  $n$ -wymiarowe wektory wejściowe  $\mathbf{X}$  oraz  $k$ -wymiarowe wektory wyjściowe  $\mathbf{Z}$  – stanowiące zestawy wymaganych sygnałów wyjściowych z elementów terminalnych czyli domykających sieć. Jeśli wektor  $\mathbf{Y}^{(j)}$  sygnałów wyjściowych z tych elementów nie będzie odpowiadał wymaganiom, to znaczy jeśli odnotujemy błąd wyrażający się różnicą wektorów  $\mathbf{Z}^{(j)} - \mathbf{Y}^{(j)}$ , to nie będziemy w stanie ustalić, w jakim stopniu za pojawienie się tego błędu odpowiadają neurony warstwy wyjściowej, na których ten błąd się ujawnił, a w jakim stopniu błąd powstał w elementach wcześniejszej warstwy, których sygnały podawane były jako wejściowe do neuronów ocenianej warstwy.

Przez wiele lat nie znano metody skutecznego uczenia sieci wielowarstwowych, a warstwy, dla których nie można było wyznaczyć sygnału błędu znane były w literaturze pod nazwą „warstw ukrytych” (ang. hidden layers). Dopiero w połowie lat 80-tych zaproponowany został algorytm wstępnej propagacji błędów (ang. backpropagation) [3], polegający na tym, że mając wyznaczony błąd  $\delta^{(m)(j)}$  występujący podczas realizacji  $j$ -tego kroku procesu uczenia w neuronie o numerze  $m$  – możemy „rzutować” ten błąd wstecz do wszystkich tych neuronów, których sygnały stanowiły wejścia dla  $m$ -tego neuronu.

Opiszemy dokładniej ten ważny algorytm. Zaczniemy od sprawy oznaczeń. W sieci wielowarstwowej niemożliwe jest ścisłe rozgraniczenie sygnałów wejściowych i wyjściowych, ponieważ sygnały wyjściowe z neuronów wcześniejszej warstwy stają się zwykle sygnałami wejściowymi dla neuronów następnej warstwy. Z tego względu wprowadzimy jednolitą numerację wszystkich neuronów (bez dzielenia ich na wejściowe, wyjściowe i należące do warstw ukrytych) oraz zastosujemy stałe oznaczenie  $y_m^{(j)}$  dla sygnału pojawiającego się na wyjściu  $m$ -tego neuronu w  $j$ -tym kroku procesu uczenia. Oczywiście niektóre spośród tych  $y_m^{(j)}$  reprezentować będą sygnały wyjściowe z całej sieci, możliwe do skonfrontowania z odpowiednimi składowymi wektora  $\mathbf{Z}^{(j)}$  w celu stwierdzenia, czy sieć pracuje poprawnie, czy też ma miejsce błąd. Podobnie uznamy, że sygnały wejściowe, dotychczas oznaczane  $\mathbf{X}$ , będą także rozpatrywane jako sygnały postaci  $y_m^{(j)}$ , czyli sygnały występujące na wyjściach wyróżnionej grupy neuronów wejściowych.

Jak z tego wynika, wszystkie sygnały w rozważanej sieci mają postać  $y_m^{(j)}$  i jedynie na podstawie numeru neuronu  $m$  można będzie rozróżnić, z jaki sygnałem (wejściowym, wyjściowym lub może jednym z sygnałów warstwy ukrytej) mamy w tym przypadku do czynienia. Dla skrócenia zapisów i zwiększenia czytelności

dalszych rozważań wprowadzimy jednak wygodną notację opartą na wyróżnieniu w zbiorze numerów neuronów  $\mathcal{M}$  ( $\forall_m m \in \mathcal{M}$ ) następujących podzbiorów:  $\mathcal{M}_x$  – zbiór numerów neuronów wejściowych do sieci;  $\mathcal{M}_y$  – zbiór numerów neuronów wyjściowych z sieci;  $\mathcal{M}_u$  – zbiór numerów neuronów warstwy ukrytej sieci;  $\mathcal{M}_i$  – zbiór numerów neuronów dostarczających sygnałów wejściowych do konkretnego, rozważanego aktualnie neuronu;  $\mathcal{M}_o$  – zbiór numerów neuronów do których dany, rozważany aktualnie neuron, wysyła swój sygnał wyjściowy.

Korzystając z tych oznaczeń możemy teraz wprowadzić zasadę uczenia wie-lowarstwowej sieci z użyciem algorytmu wstecznego rzutowania. Zaczniemy od stwierdzenia, że w  $j$ -tym kroku procesu uczenia sygnał na wyjściu każdego  $m$ -tego neuronu sieci może być wyznaczony z zestawu zależności o ogólnej postaci

$$y_m^{(j)} = \varphi \left( \sum_{i \in \mathcal{M}_i} w_i^{(m)(j)} y_i^{(j)} \right),$$

gdzie przez  $w_i^{(m)(j)}$  oznaczono wartość współczynnika wagowego synapsy łączącej wejście  $m$ -tego neuronu z wyjściem  $i$ -tego neuronu – oczywiście podczas  $j$ -tego kroku procesu uczenia. Podany wyżej wzór jest potencjalnie niejednoznaczny, ponieważ symbole typu  $y_i^{(j)}$  występują w nim po lewej i po prawej stronie rozważanej formuły, można więc wyobrazić sobie sytuację, że dla pewnych  $i_1$  i  $i_2$  dla wyznaczenia  $y_{i_1}^{(j)}$  potrzebne będzie  $y_{i_2}^{(j)}$  i z kolei dla obliczenia wartości  $y_{i_2}^{(j)}$  potrzebne będzie  $y_{i_1}^{(j)}$ . Jest to jednak niejednoznaczność pozorna, gdyż przy sieciach heteroasocjacyjnych (ang. feedforward) zawsze można wyznaczyć taką kolejność obliczania wartości  $y_m^{(j)}$ , która gwarantuje, że w momencie wyliczania wartości  $y_m^{(j)}$  dla kolejnego neuronu wszystkie wartości  $y_i^{(j)}$  dla  $i \in \mathcal{M}_i$  będą już znane. Sprawa nie jest specjalnie trudna, wystarczy tylko kolejność obliczeń związać z kolejnością przekazywania sygnałów z wejścia sieci do neuronów w kolejnych warstwach: najpierw oblicza się  $y_m^{(j)}$  dla  $m \in \mathcal{M}_x$ , potem dla  $m \in \mathcal{M}_u$  (kolejno od wejścia w kierunku wyjścia), wreszcie na końcu obliczane są wartości dla  $m \in \mathcal{M}_y$ . Sprawa się oczywiście komplikuje, jeśli w sieci występują sprzężenia zwrotne, jednak ten przypadek będzie tu pominięty.

Określony porządek jest także wymagany przy znajdowaniu wartości poprawionych wag synaptycznych, z tym, że jest on przy metodzie wstecznego rzutowania dokładnie przeciwny do tego, jaki jest wymagany przy obliczaniu wartości sygnałów  $y_m^{(j)}$  w kolejnych elementach sieci. Na samym początku wyznacza się zatem poprawki dla neuronów stanowiących wyjściową warstwę sieci ( $m \in \mathcal{M}_y$ ). Tu sprawa jest prosta, ponieważ dla poszczególnych sygnałów  $y_m^{(j)}$  istnieją w ciągu uczącym wzorcowe (oczekiwane) wartości  $z_m^{(j)}$ , z którymi można je porównywać,

wyznaczając bezpośrednio błąd  $\delta_m^{(j)}$ . Zakładamy przy tym dla prostoty, że numeracja składowych wektora wzorców  $Z^{(j)}$  jest identyczna z numeracją neuronów tworzących wyjściową warstwę sieci (tylko przy zachowaniu tego warunku  $m$  w  $y_m^{(j)}$  i  $m$  w  $z_m^{(j)}$  mogą być utożsamiane!). Wówczas

$$\Delta w_i^{(m)(j)} = \eta \delta_m^{(j)} \frac{d\varphi(e)}{de_m^{(j)}} y_i^{(j)},$$

gdzie dla  $m \in \mathcal{M}_y$

$$\delta_m^{(j)} = z_m^{(j)} - y_m^{(j)}.$$

Wzór ten dla  $m \in \mathcal{M}_y$  oraz dla  $\varphi(e)$  w postaci funkcji logistycznej zapisany może być w postaci

$$\Delta w_i^{(m)(j)} = \eta (z_m^{(j)} - y_m^{(j)}) (1 - y_m^{(j)}) y_i^{(j)} y_m^{(j)}$$

dogodnej do praktycznych obliczeń.

Na razie wprowadzane wzory nie wnoszą istotnej nowości; ich zapis jest trochę bardziej złożony (przez konieczność uwzględniania numerów  $m$ ), jednak ich ogólna struktura jest identyczna z wcześniej wprowadzonymi wzorami opisującymi proces uczenia dla pojedynczego neuronu. Jednak z chwilą skoncentrowania uwagi na neuronach warstw ukrytych sytuacja się zmienia. Można bowiem przez analogię zapisać także dla tych neuronów regułę

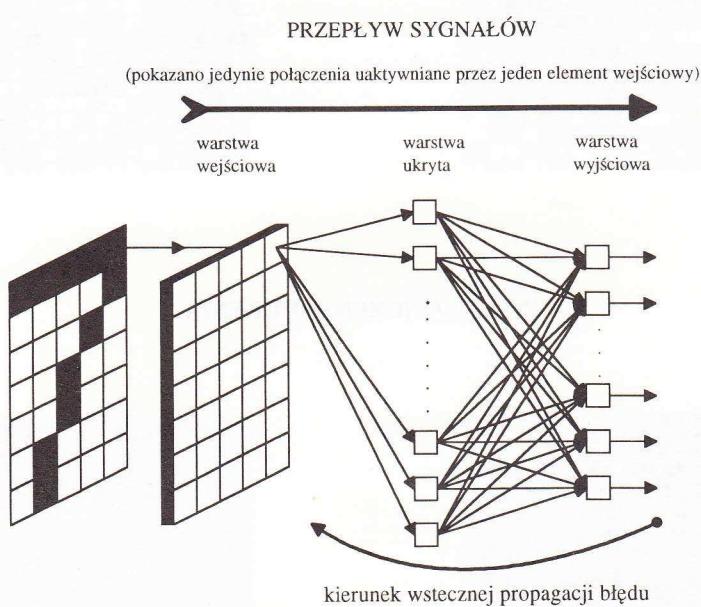
$$\Delta w_i^{(m)(j)} = \eta \delta_m^{(j)} \frac{d\varphi(e)}{de_m^{(j)}} y_i^{(j)},$$

jednak dla  $m \in \mathcal{M}_u$  nie ma możliwości bezpośredniego określenia wartości  $\delta_m^{(j)}$ . Rozważmy jednak zbiór  $\mathcal{M}_o$ . Zgodnie z wprowadzonymi oznaczeniami jest to zbiór neuronów, do których dociera sygnał  $y_m^{(j)}$  czyli sygnał wyjściowy z rozważanego neuronu warstwy ukrytej. Założymy na chwilę, że  $\mathcal{M}_o \subseteq \mathcal{M}_y$ , to znaczy założymy, że rozważany neuron należy wprawdzie do warstwy ukrytej, ale jego sygnał wyjściowy dociera wyłącznie do neuronów warstwy wyjściowej, czyli tych, dla których wartości błędów  $\delta_k^{(j)}$  mogą być bez trudu określone. Przy tym założeniu można wykazać [3], że błąd  $\delta_m^{(j)}$  neuronu warstwy ukrytej może być obliczony poprzez wsteczne rzutowanie błędów wykrytych w warstwie odbierającej sygnały:

$$\delta_m^{(j)} = \sum_{k \in \mathcal{M}_o} w_m^{(k)(j)} \delta_k^{(j)}.$$

Warto zwrócić uwagę na współczynniki wagowe, wykorzystywane przy wstecznym rzutowaniu błędów. Czynnik  $w_m^{(k)(j)}$  należy rozumieć jako wagę wejścia  $m$  neuronu  $k$  odbierającego sygnał od aktualnie rozważanego neuronu. Oznacza to, że rzutowane wstecznie błędy mnożone są przez te same współczynniki, przez które mnożone były przesyłane sygnały, tyle tylko, że kierunek przesyłania informacji zostaje w tym przypadku odwrócony: zamiast od wejścia do wyjścia przesyła się je od wyjścia kolejno w kierunku wejścia.

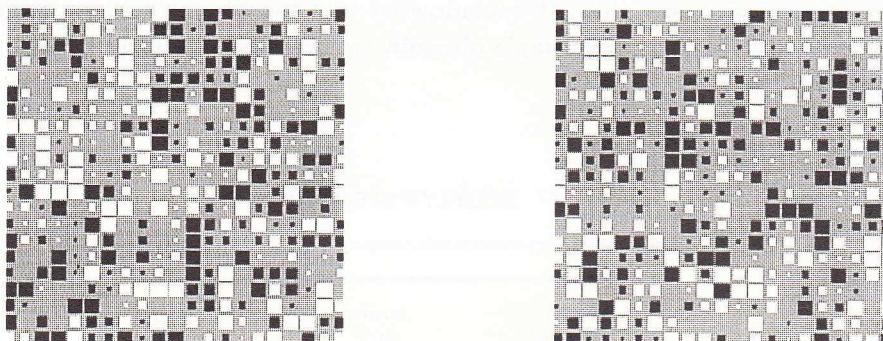
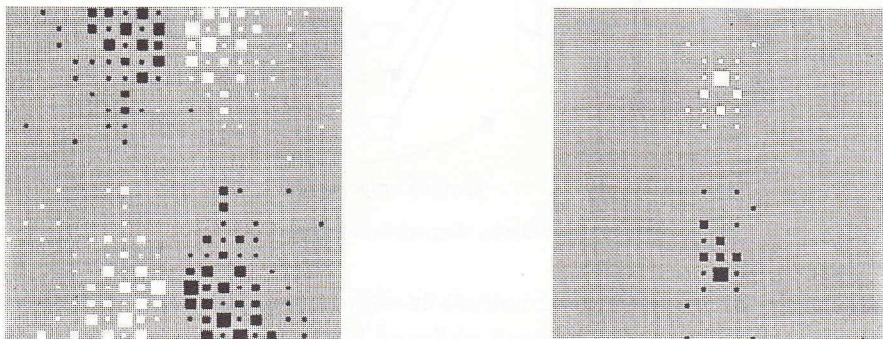
Stosując opisaną wyżej technikę wstecznej propagacji błędów można nauczyć całą sieć, ponieważ każdy neuron warstwy ukrytej albo znajduje się w przedostatniej warstwie sieci i przesyła swoje sygnały do neuronów wyjściowych (wtedy jego błąd może być wyznaczony wyżej podaną metodą) albo znajduje się w jednej z głębszej ukrytych warstw, podając sygnały do neuronów innych warstw ukrytych – wtedy jego błąd można oszacować z chwilą obliczenia błędów w neuronach będących odbiorcami jego sygnałów. Przykładowo w taki sposób uczona była używana przez autora do badań rozpoznawania obrazów trójwarstwowa sieć pokazana schematycznie na rys. 6.



Rys. 6. Struktura eksperymentalnej sieci

Okazuje się, że dla takiej sieci zawsze da się wyznaczyć taką kolejność wstecznej propagacji błędów, która pozwoli obliczyć błędy  $\delta_m^{(j)}$  dla wszystkich elementów sieci, a tym samym pozwoli znaleźć wszystkie wartości  $w_i^{(m)(j+1)}$  na podsta-

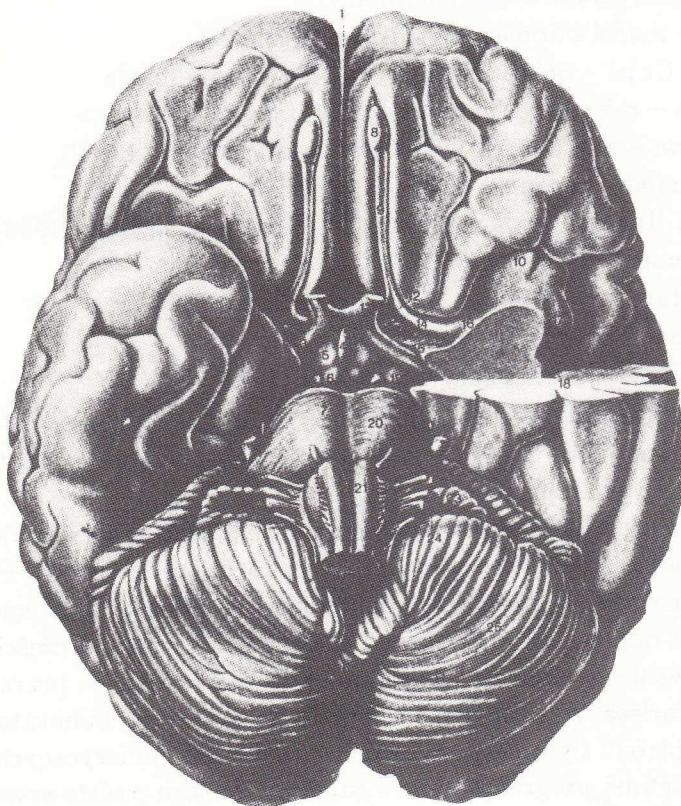
wie wartości  $w_i^{(m)(j)}$  i występujących w sieci sygnałów. Przebieg procesu uczenia przykładowej sieci przedstawiono na rys. 7, na którym pokazano współczynniki wagowe połączeń od siatkówki wejściowej do dwóch wybranych neuronów warstwy ukrytej. Wartości współczynników wagowych symbolizowane są na rysunku przez powierzchnię kwadratów, przy czym kwadraty wypełnione na czarno oznaczają wagi dodatnie, a puste – wagi ujemne. Widać jak chaotyczny pierwotnie rozkład wag zamieniany jest podczas uczenia na uporządkowany obraz celowych połączeń. Doświadczenie wskazuje, że uczenie wykonywane opisaną metodą jest stosunkowo skuteczne, ale bardzo powolne.

**PRZED ROZPOCZĘCIEM UCZENIA****PO ZAKOŃCZENIU UCZENIA**

Rys. 7. Efekty uczenia

### 3. Biologiczne korzenie neuroobliczeń i pierwsze sukcesy

Jak już wspomniano, pierwowzorem wszelkich sieci neuronowych jest oczywiście mózg ludzki (rys. 8), warto więc podać kilka informacji na jego temat. Mózg ma objętość  $1400 \text{ cm}^3$  i powierzchnię  $2000 \text{ cm}^2$  (kula o tej samej objętości ma zaledwie  $600 \text{ cm}^2$ !). Masa mózgu wynosi 1.5 kg. Jest to wartość średnia, gdyż są w tym zakresie duże różnice pomiędzy poszczególnymi ludźmi (np. mózgi kobiet są z reguły istotnie mniejsze, niż mężczyzna). Nie ma to jednak istotnego znaczenia, gdyż nie istnieje żaden naukowo wykazany związek między wagą mózgu a jego intelektualną sprawnością. Dodajmy zresztą znany fizjologom fakt, że na wymienioną masę mózgu w większości składa się woda ...



Rys. 8. Prototypem wszystkich sieci jest mózg ludzki

Zasadnicze znaczenie dla intelektualnych funkcji mózgu ma pokrywająca półkule kora mózgowa o grubości średnio ok. 3 mm, zawierająca  $10^{10}$  komórek nerwowych i  $10^{12}$  komórek glejowych. Liczbę połączeń między komórkami szacuje się

na  $10^{15}$  przy przeciętnym dystansie od 0.01 mm do 1 m. Komórki nerwowe wysyłają i przyjmują impulsy o częstotliwości 1 – 100 Hz, czasie trwania 1 – 2 ms, napięciu 100 mV i szybkości propagacji 1 – 100 m/s. Szybkość pracy mózgu można szacować na 10 operacji  $\times 10^{15}$  synaps  $\times 100$  Hz =  $10^{18}$  operacji/s (dla porównania można podać, że uznawana za najszybszy komputer świata maszyna Cray YMP ma szybkość  $10^{10}$  operacji/s). Dla wykonania typowej reakcji mózg realizuje nie więcej niż 100 elementarnych kroków, ponieważ czas reakcji jest nie mniejszy niż 300 ms, a czas reakcji pojedynczego neuronu wynosi 3 ms. Pojemności informacyjne kanałów zmysłów można szacować jako: wzrok – 100 Mb/s, dotyk – 1 Mb/s, słuch – 15 kb/s, węch – 1 kb/s, smak – 100 b/s.

Podstawy neuroobliczeń oparte są więc na fundamentalnych odkryciach neurobiologów. Wymieńmy takie ważniejsze odkrycia (podano badaczy, rodzaj odkrycia i datę przyznania Nagrody Nobla):

- Pawłow – model odruchu warunkowego (1904);
- Ramon y Cajal – opis struktury sieci nerwowej (1906);
- Einthoven – rejestracja biopotencjałów tkanek (1924);
- Sherrington – model nerwowego sterowania mięśni (1932);
- Bekesy – model percepcji słuchowej (1961);
- Hodgkin i Huxley – model propagacji sygnału w aksonie (1963);
- Eccles – model synapsy (1963);
- Granit i Hartline – badania mikroelektrodowe (1967);
- Katz – zasada „wszystko albo nic” (1970);
- Hubel i Wiesel – model kory wzrokowej (1981);
- Lorenz i Tinbergen – model celowego zachowania (1986).

Obecnie dziedzina ta ma jednak wyraźnie techniczny charakter i w takiej postaci została w tym artykule przedstawiona.

Pierwszym szeroko znanym przykładem zbudowanej i ciekawie działającej sieci neuropodobnej był *Perceptron* [4]. Był to układ częściowo elektromechaniczny (zmienne wag synaptyczne realizowano za pomocą potencjometrów obracanych przez odpowiednio sterowane silniki elektryczne), a częściowo elektroniczny (sumowanie pobudzeń). Zbudowany został w 1957 r. przez Franka Rosenblatta i Charlesa Wightmana w Cornell Aeronautical Laboratory. Przeznaczeniem perceptronu było rozpoznawanie znaków alfanumerycznych z procesem uczenia jako metodą programowania systemu. Oto jego podstawowe dane: 8 komórek nerwowych, 512 połączeń (struktura połączeń zadawana była losowo, co wynikało z rozważanej hipotezy badawczej), szybkość działania szacowana była na  $10^3$  przełączeń na sekundę.

Działanie perceptronu nie było zadowalające z punktu widzenia zasadniczego celu badań: układ wprawdzie uczył się rozpoznawania, jednak nie radził sobie z

bardziej złożonymi znakami, a ponadto wykazywał wrażliwość na zmianę skali rozpoznawanych obiektów, ich położenie w polu widzenia (przesunięcie, obrót) oraz zmiany kształtu. Zaletą perceptronu, obok faktu, że był on pierwszą realnie działającą imitacją sieci nerwowej, była zdolność do zachowywania poprawnego działania nawet po uszkodzeniu pewnej części jego elementów. Wraz ze zdolnością do uczenia się oraz z zagadkową na pozór możliwością uzyskania sensownego celowego działania struktury zbudowanej z elementów połączonych ze sobą w sposób autentycznie losowy - spowodowało to ogromne zainteresowanie eksperymentem perceptronowym w USA i na całym świecie, kilka wersji perceptronów modelował też na komputerach Odra i Cyber autor tej publikacji wraz ze współpracownikami.

Rozwój badań związanych z problematyką sieci neuronowych został gwałtownie zahamowany na początku lat 70-tych za sprawą książki [5]. Zniesięciła ona wielu badaczy, ponieważ dowodziła, że sieci jednowarstwowe (podobne do perceptronu lub oparte na zasadzie hamowania obocznych) mają bardzo ograniczony zakres zastosowań. Impas, który się wytworzył na blisko 15 lat, przełamała dopiero seria prac podających efektywne metody uczenia sieci wielowarstwowych oraz pokazujących, że nieliniowe sieci wielowarstwowe wolne są od ograniczeń wskazanych w książce [5].

Od połowy lat 80-tych notuje się prawdziwy wyścig, którego uczestnikami są obok laboratoriów badawczych także firmy produkujące układy elektroniczne. Osiągnięciami liczącymi się w tym wyścigu są: liczba elementów neuropodobnych, umieszczonych w sieci, liczba połączeń i szybkość działania. Dla zwartości opisu zestawimy ważniejsze elementy tego wyścigu w postaci Tabeli 1. Kolejne kolumny tabeli obejmują: nazwę neurokomputera (właśnie w połowie lat 80-tych utarła się ta nazwa), rok opracowania, liczbę elementów neuropodobnych umieszczonych w sieci, liczbę połączeń, szybkość działania (wyrażoną w liczbie przełączeń na sekundę) oraz nazwisko twórcy i jego firmę. Większość opracowań ma charakter zbiorowy, przytoczono jednak jedynie pierwsze nazwisko spośród wymienianych w publikacjach twórców.

Jako pewną ciekawostkę, chociaż perspektywicznie o dużym znaczeniu, warto odnotować pojawienie się w latach 80-tych neurokomputerów optycznych. Urządzenia te miały z początku skromne możliwości. Pierwszy z szerzej znanych, opracowany w 1984 r. przez Dimitrija Psaltisa z Kalifornijskiego Instytutu Technologicznego *Electro-optic crossbar* zawierał zaledwie 32 elementy. Ale już następny z wymienianych, *Optical resonator* zawierał  $6.4 \times 10^3$  elementów i  $1.6 \times 10^7$  połączeń oraz pracował z szybkością  $1.6 \times 10^5$  przełączeń w ciągu sekundy. Najnowszy z opisywanych w literaturze, zbudowany przez Dana Andersona z Uniwersytetu Colorado *Optical novelty filter*, zawiera już  $1.6 \times 10^4$  elementów i  $2 \times 10^6$  połączeń

oraz pracuje z szybkością  $2 \times 10^7$  przełączeń w ciągu sekundy.

Tabela 1. Rozwój neurokomputerów

Nazwa	Rok	Liczba elementów	Liczba połączeń	Szybkość	Twórca
Mark III	1985	$8 \times 10^3$	$4 \times 10^5$	$3 \times 10^5$	R. Hecht-Nielsen TRW
Neural emulator processor	1985	$4 \times 10^3$	$1.6 \times 10^4$	$4.9 \times 10^5$	C. Cruz IBM
Mark IV	1986	$2.5 \times 10^5$	$5 \times 10^6$	$5 \times 10^6$	R. Hecht-Nielsen TRW
Odyssey	1986	$8 \times 10^3$	$2.5 \times 10^5$	$2 \times 10^6$	A. Penz Tex. Inst. CRL
Crossbar chip	1986	256	$6.4 \times 10^4$	$6 \times 10^9$	L. Jackel AT&T Bell L.
Anza	1987	$3 \times 10^4$	$5 \times 10^5$	$1.4 \times 10^5$	R. Hecht-Nielsen Neurocomp. Corp.
Parallon	1987	$9.1 \times 10^4$	$3 \times 10^5$	$3 \times 10^4$	S. Bogoch Human Dev.
Anza plus	1988	$10^6$	$1.5 \times 10^6$	$6 \times 10^6$	R. Hecht-Nielsen Neurocomp. Corp.

#### 4. Przykłady zastosowań sieci neuronowych

Trudno wymienić wszystkie aktualnie spotykane zastosowania sieci neuronowych. Czasopismo *BYTE* (nr 8 (1992)) wymienia m.in. następujące zastosowania: diagnostykę układów elektronicznych, badania psychiatryczne, prognozy giełdowe, prognozowanie sprzedaży, poszukiwania ropy naftowej, interpretacja badań biologicznych, prognozy cen, analiza badań medycznych, planowanie remontów maszyn, prognozowanie postępów w nauce, typowania na wyścigach konnych, analiza problemów produkcyjnych, optymalizacja działalności handlowej,

analiza spektralna, optymalizacja utylizacji odpadów, dobór surowców, selekcja celów śledztwa w kryminalistyce, dobór pracowników, sterowanie procesów przemyślowych. W tym samym numerze opisano szczegółowo funkcjonowanie systemu opartego na wykorzystaniu sieci neuronowej o nazwie SNOOPE (System for Nuclear On-line Observation of Potential Explosive). System ten, wykonywany przez firmę SAIC (producenta neurokomputera Sigma I, procesora Delta II i obszernego oprogramowania dla potrzeb neuroobliczeń) służy na wielu amerykańskich lotniskach do kontroli bagażu pasażerów w celu wykrycia ewentualnych ładunków wybuchowych. Pierwszy egzemplarz systemu SNOOPE zainstalowano w międzynarodowych portach lotniczych w Los Angeles i San Francisco w lipcu 1988 r. Po pomyślnym wypróbowaniu (na ponad 40 tys. walizkach i pakunkach) system ten zainstalowano także w innych portach lotniczych. Analiza obrazu uzyskiwana w wyniku prześwietlenia bagażu dokonywana jest w pełni automatycznie, za pomocą procesora wykorzystującego możliwości uczenia się metodą wstępnej propagacji błędów. Szybkość działania oceniana jest na 10 bagażów na minutę.

Firma Bendix Aerospace dokonywała porównania systemu opartego na sieci neuronowej z typowym programem analizującym sygnały sonarowe. Stwierdzono, że sieć wykrywała obiekty podwodne skuteczniej i szybciej, a ponadto charakteryzowała się szybkim uczeniem: jej przystosowanie do nowych zadań wymagało zaledwie kilku godzin w porównaniu z kilkoma miesiącami potrzebnymi do analizicznego przestrojenia typowego programu klasyfikującego.

Firma Nestor zbudowała system automatycznego czytania znaków pisma japońskiego pod nazwą NestorWriter. System ten działa zadowalająco w wielu firmach amerykańskich dzięki zastosowaniu szybkich adaptacyjnych algorytmów rozpoznawania bazujących na sieciach neuronowych.

Szeroko znany jest program NETtalk dokonujący syntezy mowy na podstawie tekstu pisanej (w języku angielskim). Zastosowane podejście znane jest pod nazwą TDNN (Time Delay Neural Network) i uważane jest za najbardziej efektywną technikę analizy, syntezy, przetwarzania i rozpoznawania sygnałów zależnych od czasu (w tym także sygnału mowy).

Poprzestając na tych kilku konkretnych przykładach zachęcam Czytelnika do penetrowania dalszych zastosowań sieci neuronowych na podstawie bogatej i łatwo dostępnej obecnie literatury, a także do samodzielnego eksperymentów w tej dziedzinie, które spróbuję ułatwić podając niżej kilka wybranych wiadomości na temat dostępnego obecnie sprzętu i oprogramowania do celów samodzielnego budowy sieci neuronowych przez każdego, kto tylko zdecyduje się na ich stosowanie.

## 5. Oprogramowanie i sprzęt do modelowanie sieci neuronowych

Niżej podano informacje o niektórych programach i niektórych rozwiązańach sprzętowych przeznaczonych do samodzielnego eksperymentów z sieciami neuronowymi. Więcej programów i więcej typów neurokomputerów opisano w książce [1], tam też podano adresy i telefony producentów tego oprogramowania oraz adresy elektroniczne (e-mail) zagranicznych ośrodków naukowych prowadzących badania w zakresie sieci neuronowych lub oferujących miejsca dla studentów lub doktorantów chcących się w tej tematyce specjalizować. Tutaj odnotujemy tylko, że dostępne są m.in. następujące programy:

**Awareness** – program wprowadzający dla nowicjuszy, pozwalający na wykorzystanie 4 struktur sieci neuronowych. Pracuje w systemie MS-DOS, cena 275 \$, producent Neural Systems,

**AXON** – język „drugiej generacji” do opisu i projektowania sieci neuronowych. Stosowany jest w komputerach ANZA. Wspomaga go Neural Network Development Toolkit. Produkcja HNC, cena 1950 \$ za język i 3950 \$ za Toolkit,

**BrainMaker** – oprogramowanie do symulacji sieci neuronowych pracujące w systemie MS-DOS z szybkością  $5 \times 10^5$  przełączeń na sekundę. Zawiera 5 typów elementów (neuronów). Cena 99.95 \$, producent CSS,

**Cognitron** – system do projektowania, modelowania i uruchamiania sieci neuronowych, rozbudowana grafika, możliwość równoleglego uruchomienia wielu sieci, powiązania z Lispem, wersja Cognitron Prime współpracuje z transputerem T800; dostępna wersja dla MS-DOS (600 \$) i dla transputera (1800 \$), produkcja Cognitive Software,

**DynaMind** – pakiet oprogramowania dla IBM PC AT/386/486 przeznaczony do modelowania sieci i wypracowywania programów implementujących sieci. Wykorzystuje bibliotekę programów w C o nazwie NeuroLink, ma wbudowane schematy trzech popularnych sieci. Modeluje sieci o liczbie wejść do 8000 i liczbie neuronów w warstwie do 5000 (limit liczby warstw wynika z dostępnej pamięci PC). Bogata grafika. Cena 145 \$ (495 \$ w wersji DynaMind Developer emulującej chip 80170NX Intel). Producent NeuroDynamix,

**ExploreNet** – pakiet programów modelujący 19 różnych schematów sieci neuronowych z interfejsem NetSet i biblioteką modułów współpracujących UISL (User Interface Subroutine Library). Sterowany piktogramami i przystosowany do operowania dużymi zbiorami danych (w szczególności do przetwarzania sygnałów). Odmianą programu jest KnowledgeNet. Szybkość działania obu programów można zwiększyć za pomocą procesora Balboa 860. Cena w wersji MS-DOS 995 \$, w wersji VAX/VMS 3950 \$, produkcja HNC,

**MATLAB Neural Network Toolbox** – zestaw programów do znanego

pakietu matematycznego MATLAB, pozwalających modelować i badać w tym pakiecie proste struktury sieci neuronowych i proste techniki uczenia. Producent: Mathworks Inc.,

**Neural Network Development Tools** – oprogramowanie dla projektowania i badania sieci neuronowych. Zawiera m.in. NeuralWorks Professional II: program modelujący 13 typów standardowych sieci neuronowych (z możliwością definicji dalszych, własnych), zawierający 14 reguł uczenia, 10 funkcji przejścia i 11 funkcji sumujących. Wersje MS-DOS 1495 \$, Sun-4 2995 \$. NeuralWorks Explorer: program wspomagający wstępne badania. Cena 299 \$. NeuralWork Designer Pack – pakiet pozwalający projektować i badać sieci (za pomocą pakietu Professional II), produkujący po zakończeniu prób źródłowy program w C, będący prototypem użytkowego systemu (z dobrze zdefiniowanym interfejsem użytkownika) zawierającego wymaganą sieć. Cena 1995 \$. Producent: NeuralWare,

**Neuralyst for Excel** – pakiet programów pozwalających wykorzystywać do modelowania sieci neuronowych znany arkusz kalkulacyjny Excel firmy Microsoft. Praca badacza wspomagana jest przez bibliotekę Neuralyst Run-Time Library. Producent EPIC Systems Corp.,

**Savvy** – zestaw bibliotek podprogramów (w języku C dla systemu MS-DOS i dla systemu VAX/VMS) wykorzystujących technikę sieci neuronowych do rozwiązywania rzeczywistych problemów. W skład zestawu wchodzą m.in. biblioteki: Savvy Text Retrieval System, Savvy Signal Recognition System, Savvy Vision Recognition System. Cena do ustalenia z producentem, którym jest Excalibur.

Większą sprawność przetwarzania można uzyskać stosując specjalistyczne komputery lub wykorzystując specjalne układy scalone produkowane obecnie do zastosowań neuronowych. Oto niektóre z nich:

**ANZA** – Koprocesor do neuroobliczeń, tworzący z komputera IBM PC 486 specjalizowaną stację roboczą. Dostępne są dwie wersje dla IBM AT (ANZA 7000 \$ i ANZA Plus 12 500 \$) oraz ANZA Plus/VME za 24 950 \$. Produkcja ANZA,

**Balboa 860** – koprocesor przystosowany do neuroobliczeń, zbudowany w oparciu o procesor Intel 860 RISC + 16 MB cache RAM. Szybkość 40 MIPS, 80 MFLOPS,  $25 \times 10^6$  przełączeń na sekundę. Cena 10 950 \$, producent HNC,

**CMAC** – procesor neuronowy stosowany jako karta do PC-AT. Nazwa pochodzi od Cerebellar Model Arithmetic Computer. Działanie oparte na technice *look-up table*. Wykorzystywany do sterowania robotów. Cena 7950 \$, producent Shenandoach,

**Intelligent Pattern Recognition Chips** – Sieć ma możliwość pamiętania macierzy wag  $1000 \times 64$  i mnoży je przez wektor wejściowy. Cena 500 \$, producent Oxford Computer,

**Neuro-07** – komputer ten (sprzedawany wyłącznie w Japonii w cenie 11 tys. \$) pracuje z szybkością 216 000 przełączeń na sekundę. Producent NEC,

**NiSPNT404** – układ scalony, będący pierwszym procesorem mającym w swojej podstawowej liście rozkazów polecenia typowo „neuronowe” (stąd nazwa NiSP: *Neural instruction Set Processor*). Zbudowany w architekturze RISC (16 bitów, 40 MHz) osiąga wydajność 155 MIPS modelując dowolną sieć neuronową o rozmiarach do 8192 neuronów (65 536 synaps). Może współpracować z PC poprzez blok RTS (Run Time System). Za pomocą PC można rozwijać oprogramowanie systemu stosując DSS (Design System Software). Producent Neural Technologies,

**Sigma I Neurocomputer Workstation** – IBM PC 386 wraz z procesorem Delta II karta EGA (Sigma II karta VGA), producent SAIC, cena 31 500 \$. Dostępne są także: Sigma II Neurocomputer Applications Workstation (39 900 \$) i Sigma III Neurocomputer Applications Workstation (42 500 \$). Producent SAIC.

## 6. Zakończenie

Przedstawiony przegląd problematyki sieci neuronowych traktować można w najlepszym razie jako elementarne wprowadzenie do tej dziedziny. Nie omówiono wcale zagadnień samouczenia i samoorganizacji, występujących w rozważanych sieciach, nie przedstawiono problematyki sieci Hopfielda, nie pokazano związków między termodynamiką Boltzmanna a procesami uczenia sieci neuronowych. Zagadnienia te (a także inne) przedstawione są jednak w łatwo dostępnych książkach (np. [1]), zatem jeśli Czytelnik uzna na podstawie tego artykułu problematykę sieci neuronowych za interesującą i godną uwagi – może bez trudu poszerzyć i uzupełnić wiadomości szczegółowe, mając już pewien obraz całości problematyki i wstępny pogląd na temat jej ważności i przydatności.

## Literatura

- [1] R. Tadeusiewicz, *Sieci neuronowe*, wyd. II (Akademicka Oficyna Wydawnicza RM, Warszawa 1993).
- [2] R. Tadeusiewicz, „Sieci neuronowe – przewodnik problemowy”, *Elektrotechnika*, nr 2 (1991).
- [3] D. Rumelhart, „Neural Networks – a Parallel Distributed Processing Perspective”, *Proc. Neural Networks Summer School – Theory, Design and Applications* (Cambridge 1991).
- [4] F. Rosenblatt, *The perceptron. A theory of statistical separability in cognitive system*, Cornell Aeronautical Lab. Inc. Rep. No. VG-1196-G-1, 1968.
- [5] M. Minsky, S. Papert, *Perceptrons* (MIT Press, Cambridge, Mass. 1969).