Polska Akademia Nauk • Instytut Badań Systemowych

AUTOMATYKA STEROWANIE ZARZĄDZANIE

Książka jubileuszowa
z okazji
70-lecia urodzin

PROFESORA KAZIMIERZA MAŃCZAKA

pod redakcją
Jakuba Gutenbauma

Polska Akademia Nauk • Instytut Badań Systemowych

AUTOMATYKA
STEROWANIE
ZARZĄDZANIE

Książka jubileuszowa
z okazji
70-lecia urodzin

PROFESORA KAZIMIERZA MAŃCZAKA

pod redakcją
Jakuba Gutenbauma

Warszawa 2002

Książka jubileuszowa z okazji
70-lecia urodzin
Profesora Kazimierza MAŃCZAKA

Redaktor
prof. dr hab. inż. Jakub Gutenbaum

Warszawa 2002

ISBN 83-85847-78-2

Wydawca: Instytut Badań Systemowych PAN
ul. Newelska 6 01-447 Warszawa
http://www.ibspan.waw.pl

Opracowanie składopisu: Anna Gostyńska, Jadwiga Hartman

Druk: KOMO-GRAF, Warszawa
nakład 200 egz., 34 ark. wyd., 31 ark. druk.

MODELE SYSTEMÓW TWORZONE Z WYKORZYSTANIEM SIECI NEUROPODOBNYCH

Ryszard Tadeusiewicz
Akademia Górniczo-Hutnicza, Kraków, al. Mickiewicza 30

Abstract: *Neural Networks nowadays the frequently very used tools for solving problems in a wide area of applications. Every problem is different and every solution is unique in some sense. But it is possible to show all the neural solutions as the varieties of neural modeling for the problems under consideration. In the paper we try to express the most important elements of construction of such neural models for many practical and scientific problems. We try to give the most general view of the optimal selection of network architecture and the proper application of the learning process – depending on the problem under consideration and expectations connected with the solutions given by the network.*

Keywords: *neural networks, modeling, machine learning, backpropagation, approximation.*

1. Wprowadzenie

Niezmiernie popularne i chętnie stosowane sieci neuropodobne (znane także pod nazwą sieci neuronowych lub systemów konekcyjnych), są nowym i zdecydowanie ciekawym narzędziem informatycznym, przeznaczonym do przetwarzania różnych sygnałów i do analizy różnych danych. Systemy te mimo wielu lat intensywnej eksploatacji nie doczekały się nadal jednolitej i konstruktywnej teorii, takiej na przykład, która pozwoliła by na zbudowanie ogólnej metodologii ich użycia (np. odpowiadając na pytanie, jaka struktura sieci i jaki jej rozmiar powinien być zastosowany do rozwiązania każdego konkretnego problemu) oraz takiej, która uzasadniała by teoretycznie obserwowane fenomeny zachowania sieci (na przykład tego, że pozornie identycznych zadaniach raz proces uczenia pozwala na stworzenie neuronowego rozwiązania badanego problemu, a innym razem mimo wielu wysiłków nie udaje się uzyskać sukcesu). Jak podaje Eremin (2001), jednym z powodów tego stanu rzeczy jest fakt, że

w literaturze dyskutującej metody i techniki tworzenia i stosowanie sieci neuronowych opisuje się wiele ich typów oraz wiele różnych szczegółowych podejść do różnych konkretnych zadań (patrz na przykład tabela 1), podczas gdy dla budowy jednolitej teorii niezbędna jest próba unifikacji.

Tabela 1: Zestawienie typów sieci neuropodobnych.

Typ sieci	Zalety	Wady
Sieci liniowe	Proste, szybki proces uczenia, przewidywalne zachowanie	Nie pozwalają budować modeli nieliniowych
Sieci MLP	Zwarta budowa, łatwo dostępne gotowe programy	Bardzo powolny proces uczenia, częste niepowodzenia uczenia
Sieci RBF	Szybko się uczą, dobrze interpolują skomplikowane dane	Duże rozmiary, nie dokonują ekstrapolacji
Sieci Kohonena	Porządkują i samo-organizują wielowymiarowe dane	Proces samoorganizacji jest słabo kontrolowany, trudna interpretacja
Sieci Hopfielda	Pozwalają na kojarzenie danych i optymalizację	Ze względu na obecność sprzężeń zwrotnych skomplikowana dynamika
Sieci bayesowskie (probabilistyczne)	Bardzo szybki proces uczenia, dobre podstawy teoretyczne	Bardzo duże rozmiary, powolne działanie, nie dokonują ekstrapolacji

Artykuł ten oczywiście nie stworzy podwalin potrzebnej teorii, zawiera jednak propozycję, by jako czynnik unifikujący podejście do problematyki sieci neuropodobnych – przyjąć pojęcia **modelu** i modelowania. Pojęcia te wyjątkowo dobrze nadają się do tego, by stanowić dogodny punkt wyjścia do stworzenia przyszłej teorii obliczeń neuronowych. Wynika to z faktu, że sieci neuropodobne, same będąc bardzo uproszczonymi modelami fragmentów biologicznych systemów nerwowych, są jednocześnie doskonałym narzędziem właśnie do tworzenia modeli różnych innych systemów (biologicznych, technicznych, ekonomicznych, społecznych). Możliwość stosunkowo łatwego dostosowania neuronowego modelu do cech modelowanego procesu lub zjawiska w trakcie procesu **uczenia** powoduje, że ilekroć chcemy skorzystać z obliczeń neuronowych w istocie budujemy neuronowy model rozważanego problemu – chociaż nie zawsze towarzyszy temu świadoma refleksja, że to właśnie robimy.

Przedstawiona opinia może budzić wątpliwości, zwłaszcza gdy uświadomimy sobie, jak wiele różnorodnych problemów rozwiązuje się obecnie przy użyciu sieci neuropodobnych. Jest ona jednak prawdziwa. Nawet jeśli pozornie istota rozwiązywanego zadania sprowadza się do innego podejścia, nie mającego nic wspólnego z modelowaniem - to jednak twórca sieci musi na ogół tak przekształcić swój problem, by możliwe było wyróżnienie w nim dobrze zdefiniowanych danych wejściowych (przedstawianych jako pewien wejściowy wektor o strukturze i interpretacji zależnej od rozwiązywanego zadania) oraz musi swoje oczekiwania odnośnie do wyniku obliczeń także sformułować w postaci pewnego wektora (często zresztą jednoelementowego), zawierającego zestaw produkowanych przez sieć danych wyjściowych – patrz np. Toyoda i in. (2001).

Oczywiście wśród bogatej oferty różnych metod neuronowego podejścia do rozwiązywanego problemu są również takie, które nie całkowicie odpowiadają opisanej metodyce redukowania istoty rozwiązywanego problemu do samego tylko **modelowania** danych. Jako przykład sieci nie mieszczącej się w tym schemacie wymienić można sieć Hopfielda, rozważaną m.in. w pracy Wen-Jing and Lee (2001) oraz (z innych powodów) sieć Kohonena. Jednak kilka nieskomplikowanych przekształceń pozwala tak spojrzeć na problemy rozwiązywane przez te sieci, że także i w ich przypadku można mówić o pewnym neuropodobnym modelu, jako o obiekcie zainteresowania, i o modelowaniu neuronowym jako metodzie rozwiązywania formułowanych problemów.

2. Definicja sieci neuropodobnej

Zanim podejmiemy nieco dokładniej temat modelowania jako zasadniczego pojęciowego narzędzia w obliczeniach neuronowych, wyjaśnijmy (dla pełnej jednoznaczności), czym są w istocie rozważane tu sieci neuropodobne. Jak wykazał między innymi Horn (2001), ich budowie przyświeca głównie cel pragmatyczny: mają one sprawnie służyć do przetwarzania różnych sygnałów i do analizy różnych danych, a to oznacza, że powinny być jak najprostsze, bo wtedy koszt ich realizacji będzie najmniejszy. Wymóg prostoty jest uwzględniony (wręcz wbudowany w strukturę współczesnych sztucznych sieci neuropodobnych) na wiele sposobów. Po pierwsze elementy składowe tych sieci, tak zwane **sztuczne neurony**, uwzględniają jedynie trzy najprostsze właściwości swoich biologicznych odpowiedników. Według Jih-Gau Juang and Kai-Chung Cheng (2001) są to:

- zdolność do **agregacji** wielu sygnałów pochodzących z różnych źródeł (najczęściej od innych neuronów albo z wejść podających dane potrzebne do rozwiązania zadania),
- zdolność do **różnicowania** w oparciu o wiedzę gromadzoną w trakcie uczenia wpływu (na końcowy wynik obliczeń) sygnałów pochodzących z różnych źródeł,
- zdolność do **wypracowywania** sygnału wyjściowego, kierowanego do dalszych elementów sieci lub wysyłanego na zewnątrz jako składnik rozwiązania stawianego sieci zadania.

Zasada maksymalnej prostoty (w miejsce biologicznej wierności) przyświeca twórcom sztucznych sieci neuropodobnych także w momencie wyboru ich struktury. Dlatego **arbitralnie** przyjmuje się, że sztuczna sieć składać się będzie z neuronów ułożonych **warstwami**. Wyróżnia się przy tym warstwę wejściową (do której doprowadzone są sygnały stanowiące **dane** dla rozwiązywanego zadania), warstwę wyjściową (na której oczekiwać będziemy **rozwiązania** podanego przez sieć) oraz kilka (typowo od zera do trzech) warstw pośrednich, tak zwanych **ukrytych**. Taka struktura pozwala na dosyć łatwą realizację sieci, nawet w przypadku, kiedy zawiera ona dużą liczbę elementów, i to zarówno wtedy, jak sieć realizowana jest w formie programu symulacyjnego, jak i wtedy, gdy wykorzystujemy do jej realizacji specjalizowane układy (elektroniczne lub optoelektroniczne).

Neurony sieci muszą się ze sobą komunikować, bo taka jest właśnie istota działania sieci. Oznacza to, że powinny one być ze sobą **połączone**, zaś logika nakazuje, by struktura tych połączeń odpowiadała wymaganiom wynikającym z natury rozwiązywanego zadania. Problem polega tylko na tym, że sieci neuropodobnych używa się z reguły do rozwiązywania takich zadań, dla których metoda rozwiązania **nie jest** znana (wątek ten będzie dalej rozwinięty). Oznacza to jednak również brak praktycznych możliwości ustalenia, które elementy sieci powinny być ze sobą połączone, a które nie. Żeby więc nie pominąć żadnego potencjalnie potrzebnego połączenia, stosuje się w sieciach neuropodobnych nagminnie technikę organizacji połączeń typu "każdy z każdym". Takie kompletne łączenie obejmuje zwykle tylko elementy dwóch sąsiednich warstw, ale i tak liczba pierwotnie zadawanych w sieci połączeń jest bardzo duża.

Na szczęście w trakcie procesu uczenia zbyteczne połączenia (nie wnoszące żadnych pożytecznych informacji) są ignorowane, w wyniku czego po zakończonym procesie uczenia można je usunąć z "wytrenowanej" sieci. Istnieją dla realizacji tego zadania znane i wypróbowane techniki tzw.

"*pruningu*" – patrz na przykład Padhi and Balakrishnan (2001). Jednak nie zmienia to faktu, że na początku procesu uczenia trzeba brać pod uwagę wszystkie możliwe połączenia, co powoduje, że dla zamodelowania sieci zawierającej L neuronów trzeba mieć do dyspozycji L^2 miejsc w pamięci dla reprezentacji wiedzy, jaką sieć w procesie uczenia będzie gromadziła w swoich **wagach synaptycznych** (patrz dalej) związanych z występującymi w niej połączeniami. Jest to jedna z ważniejszych barier utrudniających w tym momencie tworzenie i eksploatację sieci o dużych rozmiarach.

Koszt (w tym przypadku – pamięciowy), jaki wiąże się z **wygodą** korzystania z sieci **bez** konieczności podawania jawnie sugestii dotyczących sposobu rozwiązywania rozważanego zadania – jest czymś dosyć typowym. W dalszym tekście tego artykułu spotkamy się jeszcze kilka razy z faktem, że budując sieć **możemy** nie myśleć o szczegółach jej budowy i działania, ale odbywa się to zawsze jakimś **kosztem** – najczęściej kosztem czasu obliczeń (związanych głównie z uczeniem sieci, ale do pewnego stopnia również z szybkością jej eksploatacji).

Kolejnym ważny założeniem upraszczającym, przyjmowanym bardzo często przy budowie sieci, jest założenie o jednokierunkowym charakterze przepływu informacji przez sieć. Zdecydowana większość **praktycznie** używanych sieci charakteryzuje się tym, że informacje są w nich przesyłane ściśle jednokierunkowo – od wejścia do wyjścia. Jak wskazano między innymi w pracy Toyoda i in. (2001), struktury sieci zawierających sprzężenia zwrotne należą do rzadkości, chociaż niektóre z nich (na przykład sieci Hopfielda) zyskały pewną popularność. Wynika to z faktu, że przy tworzeniu sieci neuropodobnej traktowanej jako narzędzie obliczeniowe unika się (jak tylko można) wszelkich komplikacji, a niewątpliwie sporą komplikacją są (występujące w strukturach ze sprzężeniem zwrotnym) procesy dynamiczne, prowadzące czasem do niestabilności oraz niekiedy generujące nie zanikające procesy dynamiczne w sieci – oscylacyjne lub chaotyczne.

3. Charakterystyka modeli neuropodobnych

3.1. Uwagi ogólne

W tradycyjnym podejściu do modelowania matematycznego różnych złożonych systemów (np. w modelowaniu wykorzystującym podejście statystyczne) stosowane są różne **algorytmy** określające taką konfigurację parametrów modelu, która zapewnia osiągnięcie minimum **globalnego** przez funkcję błędu (por. wzór (6)). Oznacza to, że stworzony (na przykład

z pomocą metody regresji) model matematyczny (najczęściej liniowy) jest **najlepszym** możliwym odwzorowaniem posiadanego zbioru danych, uzyskanym z pomocą modelu **założonego typu**.

Czy jest więc celowe budowanie modeli neuropodobnych, o których z góry wiadomo, że są na skutek wrażliwości metod ich uczenia na minima lokalne funkcji błędu (co będzie jeszcze dokładniej przedyskutowane), mogą modelować rozważany proces lub badane zjawisko w sposób **suboptymalny**?

Jak wykazał m.in. Horn (2001), odpowiedź jest pozytywna i opiera się na fakcie, że typowe metody modelowania matematycznego dostarczają rozwiązań najlepszych, ale tylko w a'priori określonej (arbitralnie!) klasie modeli. Na przykład model regresji liniowej **jest** modelem najlepszym – ale najlepszym wyłącznie *w klasie modeli liniowych*. Nie da się zbudować modelu *liniowego*, który by lepiej pasował do posiadanych danych niż model uzyskany za pomocą metod regresji - ale nie jest wykluczone, że istnieje model *nieliniowy*, który będzie opisywał te same dane nieporównanie lepiej – patrz Wei-Song Lin and Chun-Sheng Chen (2001). Jednak świadomość tego faktu jest mało przydatna w praktyce, ponieważ klasa modeli nieliniowych jest klasą otwartą, jako że należą do niej wszelkie modele, które nie są liniowe, a takich modeli jest nieskończenie dużo. W związku z tym znalezienie modelu nieliniowego oznacza w istocie wykonanie **dwóch** czynności: wyboru kształtu modelu (lub jego ogólnego wzoru matematycznego) oraz doboru takich parametrów tego modelu, żeby odwzorowywał on posiadane dane w optymalny sposób. Druga część zadania daje się łatwo zautomatyzować, gdyż można tu zastosować odpowiednie metody (np. statystyczną technikę regresji nieliniowej) do wyznaczenia optymalnych parametrów modelu nieliniowego. Jednak dotyczy to zawsze modelu **o uprzednio wybranym kształcie.** Zatem przed uruchomieniem wygodnego i szybkiego procesu doboru parametrów modelu (w toku jego identyfikacji) użytkownik musi **zgadnąć** poprawny kształt tworzących go równań – co w ogólnym przypadku nie jest łatwe.

Sieci neuropodobne wnoszą do procesu modelowania nową jakość. Sieć w trakcie procesu uczenia może bowiem **całkiem sama** znaleźć nieliniowy model rozważanego systemu, przy czym twórca sieci **nie musi** jej podawać żadnych założeń dotyczących kształtu tego modelu. Stosowanie modeli opartych na sieciach neuropodobnych, a zwłaszcza modeli opartych na tej technice używanych do opisu zjawisk i procesów silnie nieliniowych – zwiększa zatem **istotnie** możliwości ich modelowania.

Jednak nie należy sądzić, że sieci neuropodobne mogą być traktowanej jako *panaceum* na wszystkie problemy. Przeciwnie, biorąc pod

uwagę opisane dalej fakty dotyczące procesu uczenia sieci przyznać trzeba, że ceną płaconą za tę wygodę jest brak pewności, że w ogóle uda się zbudować jakiś sensowny model. Omówimy to nieco dokładniej, bo jest to ważna okoliczność, mająca często zasadniczy wpływ na ocenę przydatności (lub nieprzydatności) sieci neuropodobnych w kontekście określonych zadań.

Sieć neuropodobna samodzielnie nabywa umiejętności rozwiązywania stawianych jej zadań w trakcie procesu uczenia, czyli pokazywania jej przykładów zadań poprawnie rozwiązanych. Jest to bardzo wygodne, ponieważ zwykle pozwala zbudować potrzebny model w miarę prosto i bez kłopotu. Co więcej, dzięki temu sposobowi samodzielnego nabywania wiedzy przez sieci – można je obarczać także takimi zadaniami, dla których twórca sieci **nie zna** poprawnej metody ich rozwiązywania. Typowym przykładem są tu zadania klasyfikacji i rozpoznawania albo obszerna klasa zadań związanych z prognozowaniem. W istocie na zasadzie procesu uczenia połączonego z uogólnianiem zdobytej wiedzy możemy **próbować** nauczyć sieć rozwiązywania dowolnego zadania, jeśli dysponujemy dostatecznie dużą liczbą przykładów, dla których znane są zarówno zespoły przyczyn, jak i obserwacje ich skutków. Jeśli związek przyczynowy między wskazanymi wejściami i wyjściami w rozważany przypadku zachodzi, to sieć neuronowa **zazwyczaj** go wykryje, a to pozwoli na używanie neuronowego modelu jako podstawy do dalszego rozwiązywania problemu – na przykład prognozowania.

Za wygodę braku konieczności myślenia przy wykorzystywaniu neuropodobnych modeli przychodzi jednak zapłacić dosyć wysoką cenę, gdyż proces nabywania wiedzy przez sieć neuronową jest procesem zdecydowanie stochastycznym. Dalszy opis niektórych (spośród bardzo wielu znanych) metod uczenia sieci pozwoli zorientować się co do przyczyn tego indeterminizmu, w tym momencie ważny jest jednak głównie jego skutek, a skutkiem tym jest wątpliwość, z jaką stale mamy do czynienia w przypadku używania sieci, polegająca na tym, że **nigdy** tak naprawdę nie wiemy, czy w trakcie uczenia sieci poziom błędu osiągnął swój poziom minimalny, czy nie. Oznacza to, że przy niewłaściwie prowadzonym uczeniu sieci uzyskany model może być ostatecznie bardzo odległy od modelu optymalnego – i co gorsza wcale nie będziemy o tym poinformowani!

Dla konkretyzacji dalszych rozważań ustalmy, że przez model optymalny będziemy (tutaj i potem dalej) rozumieć model, jaki mógłby być uzyskany w tych samych warunkach, gdyby użytkownik **wiedział,** jaki jest najodpowiedniejszy kształt nieliniowej formuły poszukiwanego modelu, a potem stosowną procedurą optymalizacyjną (np. regresyjną) dobrał

najlepsze wartości odpowiednich parametrów na podstawie wszystkich posiadanych danych.

W dążeniu do stworzenia neuronowego modelu optymalnego musimy pokonać dwie bariery: znaleźć optymalną strukturę sieci i maksymalnie skutecznie przeprowadzić proces uczenia. Jak wykazał między innymi Salcic (2001), te dwa zadania są ze sobą związane, gdyż dla każdej struktury sieci trzeba dobrać algorytm uczenia pozwalający na maksymalne wykorzystanie atutów tej wybranej struktury.

Istnieje wiele typów i rodzajów sieci neuropodobnych, różniących się między sobą strukturą i zasadami działania. Podstawowe informacje na temat niektórych (najbardziej popularnych) spośród nich podaje tabela 1. Jest rzeczą oczywistą, że możliwości przetwarzania informacji w sieciach o różnej strukturze są silnie zróżnicowane. Stąd przedstawiając różne możliwe struktury tych sieci (a potem metody ich uczenia) będziemy też sygnalizować (zwykle jednak bardzo skrótowo), do jakich klas zadań te właśnie sieci w szczególnym stopniu się nadają.

3.2. Sieci liniowe

Rozważając proces tworzenia neuronowych modeli różnych problemów naukowych i technicznych należy stwierdzić, że najprostszym jest **zawsze** model liniowy. Model liniowy jest reprezentowany przez sieć nie posiadającą warstw ukrytych, zaś neurony znajdujące się w jej warstwie wyjściowej są w pełni liniowe (tzn. są to neurony, w których łączne pobudzenie wyznaczane jest jako liniowa kombinacja wartości wejściowych i które posiadają liniową funkcję aktywacji). W modelu takim funkcją dopasowywaną do posiadanych danych jest hiperpłaszczyzna, a dokładniej – L hiperpłaszczyzn, gdzie L jest liczbą neuronów. Zadanie polega na znalezieniu właściwego położenia i właściwego nachylenia każdej takiej hiperpłaszczyzny. Sieci neuronowe liniowe stanowią wygodne narzędzie możliwe do bezpośredniego użycia w wielu istotnych zastosowaniach (na przykład w przetwarzaniu sygnałów mogą być skutecznie używane do filtracji zakłócających szumów z automatycznym – w trakcie uczenia dobieranym – pasmem przepustowym sygnału). Sieci liniowe są szczególnie korzystne, jeśli zbiór uczący składa się tylko z niewielu przypadków (występuje brak danych potrzebnych do uczenia sieci). W takich przypadkach zastosowanie bardziej złożonych modeli nie jest w praktyce możliwe (nie uda się skutecznie nauczyć złożonej sieci przy użyciu niewielkiej liczby danych uczących).

3.3. Sieci nieliniowe

Sieci nieliniowe, których możliwości tworzenia modeli złożonych systemów są znacząco większe, niż sieci liniowych, mogą mieć różną strukturę (patrz na przykład Shao Qing i in. (2001)). Warto w związku z tym odnotować następujące uwagi na ich temat:

- sieć nie posiadająca wcale warstw ukrytych może służyć głównie do rozwiązywania problemów liniowo – separowalnych.
- Sieć z pojedynczą warstwą ukrytą pozwala wyróżnić bardziej skomplikowany obszar w przestrzeni danych wejściowych, musi to być jednak obszar **jednospójny** i **wypukły.**
- Podane wyżej ograniczenia można złagodzić poprzez odpowiednią interpretację danych (np. przez uwzględnienie, że **dopełnienie** obszaru wypukłego jest obszarem wklęsłym).
- Dodatkowe możliwości kształtowania powierzchni decyzyjnej wynikają z faktu, że obszar decyzyjny może rozciągać się aż do nieskończoności, co powoduje, że sieć dwuwarstwowa w praktyce jest zdolna do modelowania we właściwy sposób **większości** rzeczywistych problemów klasyfikacyjnych – chociaż nie wszystkich.
- Dopiero sieć trójwarstwowa (posiadająca **dwie** warstwy ukryte) ma na tyle bogaty asortyment możliwych zachowań, że może modelować wszystkie (bez wyjątku) rodzaje zależności danych wyjściowych od danych wejściowych.

W praktyce większość problemów wymaga użycia najwyżej pojedynczej warstwy ukrytej, a tylko w pewnych bardzo szczególnych przypadkach należy zastosować sieć posiadającą dwie warstwy ukryte. Większość specjalistów sądzi, że potrzeba zastosowania trzech warstw ukrytych nie pojawia się praktycznie nigdy, a ci, którzy stosują sieci zawierające trzy lub więcej warstw ukrytych postępują bardzo nieroztropnie – zwłaszcza gdy się uwzględni złożoność procesu uczenia zdecydowanie rosnącą z każdą warstwą sieci.

4. Metody uczenia sieci

Jak wykazano m.in. w pracy Jagannathan i Tohmaz (2001), doprowadzenie sieci neuronowej do tego, by mogła być traktowana jako model określonego problemu (i żeby dzięki temu mogła rozwiązywać te problemy) jest możliwe po zastosowaniu odpowiedniej metody uczenia. Ze względu na to, że uczenie ma zawsze kluczowe znaczenie dla tworzenia

neuronowych modeli i dla ich wykorzystywania do znajdowania rozwiązań formułowanych problemów – niżej omówimy najpopularniejsze metody uczenia sieci. Opis działania rozważanych algorytmów uczenia sieci przedstawiony zostanie w oparciu o ciąg uczący o postaci:

$$\langle x_1, d_1\rangle, \langle x_2, d_2\rangle, \ldots, \langle x_R, d_R\rangle \tag{1}$$

gdzie wartości wejściowe $x_p \in R^N$, zaś oczekiwane (zakładane) wartości wyjściowe $d_p \in R^M$ określone są dla każdego $p = 1, \ldots, R$. W trakcie działania sieci neuronowej (zakładać będziemy architekturę sieci jednokierunkowej charakteryzującą się posiadaniem N wejść i M wyjść z jedną warstwą ukrytą zawierającą L neuronów) wyróżnić można następujące etapy:

a) wprowadzenie na wejście sieci p – tego wektora wejściowego (x_p),

b) obliczenie dla każdego neuronu warstwy ukrytej wartości net_{pj}^h zgodnie ze wzorem:

$$net_{pj}^h = \sum_{i=0}^{N} w_{ji}^h x_{pi} \tag{2}$$

c) obliczenie wartości wyjściowej y_{pj}^h dla każdego neuronu warstwy ukrytej:

$$y_{pj}^h = f_j^h\left(net_{pj}^h\right) \tag{3}$$

d) obliczenie dla każdego neuronu warstwy wyjściowej wartości net_{pk}^o:

$$net_{pk}^o = \sum_{j=0}^{L} w_{kj}^o y_{pj}^h \tag{4}$$

e) obliczenie wartości wyjściowej dla każdego neuronu warstwy wyjściowej:

$$y_{pk}^o = f_k^o\left(net_{pk}^o\right) \tag{5}$$

W prawidłowo działającej sieci wektor wartości wyjściowych $\mathbf{y_p}$ powinien być identyczny lub bardzo zbliżony do wektora wartości oczekiwanych $\mathbf{d_p}$. Aby do tego doprowadzić stosuje się różne algorytmy uczenia, minimalizujące błąd E_p wyrażany formuła:

$$E_p = \frac{1}{2}\sum_{k=1}^{M}\left(d_{pk} - y_{pk}^{o}\right)^2 \tag{6}$$

Najbardziej popularny jest klasyczny algorytm *wstecznej propagacji błędów,* który został zaproponowany przez D. Rumelharta, G. Hintona i R. Williamsa w 1986 roku, w którym zmiany wartości wag mają charakter iteracyjny i bazują na **metodzie największego spadku**. Po prezentacji *p*-tego elementu ciągu uczącego modyfikacja wektora wag odbywa się zgodnie ze wzorem:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla E_p\left(\mathbf{w}^{(t)}\right) \tag{7}$$

gdzie:

$\mathbf{w}^{(t)}$ – wektor wszystkich wag sieci w *t*-tym kroku działania algorytmu,

$\nabla E_p(\mathbf{w}^{(t)})$ – gradient funkcji E_p w punkcie $\mathbf{w}^{(t)}$,

η – współczynnik uczenia będący stałą z przedziału (0, 1).

W trakcie działania algorytmu modyfikowane być muszą zarówno w_{kj}^{o} neuronów warstwy wyjściowej jak i wagi w_{ji}^{h} neuronów warstwy ukrytej. Modyfikacja wag przebiega zgodnie z wzorami:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) - \eta \frac{\partial E_p}{\partial w_{kj}^{o}} \tag{8}$$

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) - \eta \frac{\partial E_p}{\partial w_{ji}^{h}} \tag{9}$$

W celu obliczenia wartości $\frac{\partial E_p}{\partial w_{kj}^{o}}$ najpierw oblicza się pochodną $\frac{\partial E_p}{\partial y_{pk}^{o}}$

$$\frac{\partial E_p}{\partial y_{pk}^{o}} = \frac{\partial}{\partial y_{pk}^{o}}\left(\frac{1}{2}\sum_{k=1}^{M}\left(d_{pk} - y_{pk}^{o}\right)^2\right) = -\left(d_{pk} - y_{pk}^{o}\right) \tag{10}$$

następnie

$$\frac{\partial E_p}{\partial net^o_{pk}} = \frac{\partial E_p}{\partial y^o_{pk}} \frac{\partial y^o_{pk}}{\partial net^o_{pk}} = -\left(d_{pk} - y^o_{pk}\right) \frac{\partial \mathrm{f}\left(net^o_{pk}\right)}{\partial net^o_{pk}} \tag{11}$$

a po wprowadzeniu dodatkowej zmiennej:

$$\delta^o_{pk} = \left(d_{pk} - y^o_{pk}\right) \frac{\partial \mathrm{f}\left(net^o_{pk}\right)}{\partial net^o_{pk}} \tag{12}$$

zależność (11) można uprościć do postaci:

$$\frac{\partial E_p}{\partial net^o_{pk}} = -\delta^o_{pk} \tag{13}$$

Kolejnym krokiem jest obliczenie wpływu zmiany wagi w^o_{kj} na wartość E_p:

$$\frac{\partial E_p}{\partial w^o_{kj}} = \frac{\partial E_p}{\partial net^o_{pk}} \frac{\partial net^o_{pk}}{\partial w^o_{kj}} =$$

$$= -\delta^o_{pk} \frac{\partial}{\partial w^o_{kj}} \left(w^o_{k0} + w^o_{k1} y^h_{p1} + \ldots + w^o_{kj} y^h_{pk} + \ldots + w^o_{kL} y^h_{pL}\right) = -\delta^o_{pk} y^h_{pk}$$

i ostatecznie:

$$w^o_{kj}(t+1) = w^o_{kj}(t) + \eta \delta^o_{pk} y^h_{pj} \tag{14}$$

W analogiczny sposób wyznaczyć można formułę określającą zmiany wag w neuronach warstwy ukrytej:

$$\frac{\partial E_p}{\partial y^h_{pj}} = \frac{\partial}{\partial y^h_{pj}} \left(\frac{1}{2} \sum_{k=1}^{M} \left(d_{pk} - y^o_{pk}\right)^2 \right) = \sum_{k=1}^{M} \frac{\partial}{\partial y^h_{pj}} \left(\frac{1}{2} \left(d_{pk} - y^o_{pk}\right)^2 \right) =$$

$$= -\sum_{k=1}^{M} \left(d_{pk} - y^o_{pk}\right) \frac{\partial \mathrm{f}\left(net^o_{pk}\right)}{\partial y^h_{pj}} = -\sum_{k=1}^{M} \left(d_{pk} - y^o_{pk}\right) \frac{\partial f\left(net^o_{pk}\right)}{\partial net^o_{pk}} \frac{\partial net^o_{pk}}{\partial y^h_{pj}} =$$

$$= -\sum_{k=1}^{M} \left(d_{pk} - y^o_{pk}\right) \frac{\partial \mathrm{f}\left(net^o_{pk}\right)}{\partial net^o_{pk}} w^o_{kj} = -\sum_{k=1}^{M} \delta^o_{pk} w^o_{kj} \tag{15}$$

Następnie:

$$\frac{\partial E_p}{\partial net_{pj}^h} = \frac{\partial E_p}{\partial y_{pj}^h}\frac{\partial y_{pj}^h}{\partial net_{pj}^h} = -\left(\sum_{k=1}^{M}\delta_{pk}^o w_{kj}^o\right)\frac{\partial \mathrm{f}\left(net_{pj}^h\right)}{\partial net_{pj}^h} \tag{16}$$

i po wprowadzeniu dodatkowej zmiennej:

$$\delta_{pj}^h = \left(\sum_{k=1}^{M}\delta_{pk}^o w_{kj}^o\right)\frac{\partial \mathrm{f}\left(net_{pj}^h\right)}{\partial net_{pj}^h} \tag{17}$$

równanie (16) zapisać można w postaci:

$$\frac{\partial E_p}{\partial net_{pj}^h} = -\delta_{pj}^h \tag{18}$$

i wtedy:

$$\frac{\partial E_p}{\partial w_{ji}^h} = \frac{\partial E_p}{\partial net_{pj}^h}\frac{\partial net_{pj}^h}{\partial w_{ji}^h} = -\delta_{pj}^h x_{pi} \tag{19}$$

Mając obliczoną wartość pochodnej z równania (19) można już zapisać wzór na zmianę wag dla neuronów warstwy ukrytej:

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \eta\delta_{pj}^h x_{pi} \tag{20}$$

W celu przezwyciężenia wad algorytmu *backpropagation*, takich jak:

- powolność algorytmu,
- tendencja do zatrzymywania się algorytmu w minimach lokalnych

stosuje się czasem *uczenie skumulowane* polegające na tym, że po prezentacji każdego wzorca obliczany jest wektor zmiany wag Δw_p, ale modyfikacja wartości wag dokonywana jest po prezentacji wszystkich elementów ciągu uczącego. Wagi zmieniane są wtedy o wartość:

$$\Delta\mathbf{w} = \sum_{p=1}^{R}\Delta\mathbf{w}_p \tag{21}$$

W tym samym celu wprowadza się niekiedy dodatkowy składnik nazywany pędem *(momentum)*:

$$\Delta\mathbf{w}^t = -\eta\nabla E\left(\mathbf{w}^t\right) + \mu\Delta\mathbf{w}^{t-1} \tag{22}$$

gdzie μ jest współczynnikiem określającym wpływ poprzedniej modyfikacji wektora wag na aktualny wektor poprawek. Innym zabiegiem stosowanym w celu polepszenia jakości procesu uczenia jest to, że różnicuje się współczynniki uczenia związane z poszczególnymi składowymi wektora wag:

$$\Delta w_i^{(t)} = -\eta_i^{(t)} \cdot \nabla_i E\left(w^t\right) \tag{23}$$

W początkowej fazie algorytmu wszystkim współczynnikom uczenia nadaje się taką samą wartość, będącą niewielką liczbą dodatnią $\eta_i^{(0)}$. Po wykonaniu kroku t dokonuje się zmiany wartości współczynnika uczenia w następujący sposób:

$$\eta_i^{(t+1)} = \begin{cases} \eta_i^{(t)} \cdot u & jeżeli \quad \nabla_i E\left(w^t\right) \cdot \nabla_i E\left(w^{t-1}\right) \geq 0 \\ \eta_i^{(t)} \cdot d & jeżeli \quad \nabla_i E\left(w^t\right) \cdot \nabla_i E\left(w^{t-1}\right) < 0 \end{cases} \tag{24}$$

gdzie wartości u oraz d są odpowiednio dobranymi stałymi spełniającymi warunki: $u > 1$ oraz $d < 1$. Wadą przedstawionej we wzorze (24) formuły modyfikacji wag jest szybki, wykładniczy wzrost wartości współczynnika uczenia w sytuacji, gdy przez kilka kolejnych iteracji znak odpowiedniego elementu wektora gradientu nie ulega zmianie. Wady tej nie posiada inna reguła modyfikacji współczynnika uczenia:

$$\eta_i^{(t+1)} = \eta_i^{(t)} - \gamma \frac{\partial E\left(w^t\right)}{\partial \eta_i^{(t)}} \tag{25}$$

gdzie γ jest pewnym dodatnim, stałym współczynnikiem z przedziału [0, 1]. Można wykazać, że wzór (25) jest równoważny formule:

$$\eta_i^{(t+1)} = \eta_i^{(t)} + \gamma \frac{\partial E\left(w^t\right)}{\partial w_i^t} \cdot \frac{\partial E\left(w^{t-1}\right)}{\partial w_i^{t-1}} \tag{26}$$

Ze wzoru (26) wynika, że współczynnik uczenia rośnie, jeżeli odpowiednie składowe wektora gradientu nie zmieniają znaku w dwóch kolejnych iteracjach. W przeciwnym przypadku współczynnik uczenia maleje.

Reguła adaptacji współczynnika uczenia dana wzorem (26) nosi nazwę *delta - delta*. Wskazać można na jej dwie podstawowe wady: duża wrażliwość na zmiany składowych wektora gradientu oraz trudności we właściwym doborze wartości współczynnika γ. Z tego powodu w praktyce stosuje się często podaną przez Jacobsa modyfikację reguły (26) o postaci:

$$\eta_i^{(t+1)} = \begin{cases} \eta_i^{(t)} + u & jeżeli \quad \nabla_i E\left(w^t\right) \cdot \delta_i^{(t-1)} > 0 \\ \eta_i^{(t)} \cdot d & jeżeli \quad \nabla_i E\left(w^t\right) \cdot \delta_i^{(t-1)} < 0 \\ \eta_i^{(t)} & jeżeli \quad \nabla_i E\left(w^t\right) \cdot \delta_i^{(t-1)} = 0 \end{cases} \tag{27}$$

gdzie u oraz d są wartościami stałymi, zaś $\delta_i^{(t)}$ dane jest wzorem:

$$\delta_i^{(t)} = (1-\phi)\nabla_i E\left(w^t\right) + \phi\delta_i^{(t-1)} \tag{28}$$

gdzie ϕ jest pewną stałą dodatnią. Reguła (27) - (28) nosi nazwę *delta - bar - delta*. Powiększanie wartości współczynnika uczenia następuje w sposób liniowy poprzez dodanie stałej u, zaś zmniejszanie współczynnika dokonywane jest w sposób wykładniczy poprzez przemnożenie przez mniejszą od jedności stałą d. Liniowy wzrost zabezpiecza przed zbyt szybkim narastaniem, zaś wykładniczy spadek uniemożliwia osiągnięcie wartości ujemnej. Typowe wartości dla stałych występujących we wzorze (27) wynoszą: $0 < u < 0.05$ oraz $0.1 < d < 0.3$.

Czasem stosuje się też inny mechanizm zanikania wag:

$$\Delta\mathbf{w}^t = -\eta\nabla E\left(\mathbf{w}^t\right) - \beta\mathbf{w}^t \tag{29}$$

Autorem najbardziej znanej szybkiej odmiany algorytmu wstecznej propagacji błędów (tak zwanego algorytmu *Quickprop*) jest S. E. Fahlman. W metodzie tej zakłada się, że funkcja błędu jest lokalnie paraboloidalna. Korzystając z tego założenia wartość wektora poprawki $\Delta\mathbf{w}^t$ oblicza się w taki sposób, aby w trakcie jednej iteracji osiągnąć minimum funkcji błędu (minimum paraboloidy). Wartość poprawki dana jest wzorem:

$$\Delta\mathbf{w}^t = \frac{\nabla E\left(\mathbf{w}^t\right)}{\nabla E\left(\mathbf{w}^{t-1}\right) - \nabla E\left(\mathbf{w}^t\right)} \Delta\mathbf{w}^{t-1} \tag{30}$$

Kolejna modyfikacja metody wstecznej propagacji błędów, zwana ***RPROP*** *(resilient backpropagation)*, której autorami są M. Riedmiller oraz H. Braun wykorzystuje minimalizację funkcji błędu o postaci:

$$E = \sum_{p=1}^{R}\sum_{k=1}^{M}\left(d_{pk} - y_{pk}^o\right)^2 + 10^{-\alpha}\sum w_{ij}^2 \tag{31}$$

gdzie: α – parametr określający tempo zanikania wag,

$\sum w_{ij}^2$ – suma kwadratów wszystkich wag występujących w sieci.

Modyfikacja wartości wag w algorytmie RPROP dokonywana jest po prezentacji wszystkich elementów ciągu uczącego (uczenie skumulowane). Każda z modyfikowanych wag w_{ij} występujących w sieci zmieniana jest w następujący sposób:

$$\Delta w_{ij}^{t} = \begin{cases} -\Delta_{ij}^{t}, & gdy \quad \dfrac{\partial E\left(w^{t}\right)}{\partial w_{ij}} > 0 \\ \Delta_{ij}^{t}, & gdy \quad \dfrac{\partial E\left(w^{t}\right)}{\partial w_{ij}} < 0 \\ 0, & gdy \quad \dfrac{\partial E\left(w^{t}\right)}{\partial w_{ij}} = 0 \end{cases} \tag{32}$$

gdzie Δ_{ij}^{t} jest wielkością, o którą zmienia się wartość wagi w_{ij} w t-tej epoce procesu uczenia. Po dokonaniu zmiany wag następuje proces modyfikowania wielkości Δ_{ij}. Dokonuje się on w następujący sposób:

$$\Delta_{ij}^{t+1} = \begin{cases} \theta^{+} * \Delta_{ij}^{t}, & gdy \quad \dfrac{\partial E\left(w^{t}\right)}{\partial w_{ij}} \dfrac{\partial E\left(w^{t+1}\right)}{\partial w_{ij}} > 0 \\ \theta^{-} * \Delta_{ij}^{t}, & gdy \quad \dfrac{\partial E\left(w^{t}\right)}{\partial w_{ij}} \dfrac{\partial E\left(w^{t+1}\right)}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{t}, & gdy \quad \dfrac{\partial E\left(w^{t}\right)}{\partial w_{ij}} \dfrac{\partial E\left(w^{t+1}\right)}{\partial w_{ij}} = 0 \end{cases} \tag{33}$$

gdzie $0 < \theta^{-} < 1 < \theta^{+}$.

5. Podsumowanie

Wyżej opisano (z konieczności w dużym skrócie), jak można interpretować rozwiązywanie różnych zadań z użyciem sieci neuropodobnych jako proces budowy ich modeli. Najwięcej uwagi poświęcono problemowi uczeni sieci, gdyż stanowi on w istocie klucz do wszelkich technik neuronowego modelowania. Nie było jednak celem tej pracy przedstawianie wyczerpującego studium problemu, lecz zwrócenie uwagi badaczy (szczególnie tych, którzy dopiero zaczynają swoją przygodę z sieciami neuropodobnymi) na fakt, że zbyt jednostronne ograniczanie rozważań wyłącznie do możliwości, jakie oferuje algorytm *backpropagation*, może prowadzić do niepowodzeń przy stosowaniu sieci

neuropodobnych, podczas gdy sięgnięcie do nowych metod uczenia – w istotnym stopniu zwiększa prawdopodobieństwo uzyskania zadowalających rozwiązań – i to z reguły po znacząco krótszym czasie uczenia. Przedstawione w artykule algorytmy powinny skłonić osoby stosujące sieci neuronowe do tego, by zdecydowały się spojrzeć na problem uczenia sieci w sposób mniej rutynowy i bardziej elastyczny, co z pewnością się opłaci, ponieważ – co warto na koniec jeszcze raz podkreślić – proces uczenia jest kluczem do sukcesu we wszystkich zastosowaniach sieci neuropodobnych.

Spróbujemy też w tym podsumowaniu wypowiedzieć ogólny sąd o całej technice modelowania sieci neuropodobnych. Niektórzy badacze formułują bardzo mocne twierdzenie, głoszące, iż nieliniowa sieć o dowolnej liczbie elementów i o dowolnej liczbie warstw – **może być modelem dowolnego zjawiska**, albo – wyrażając to samo w innej terminologii – że sieć **może aproksymować dowolną funkcję**. Takie "mocne" twierdzenia pobudzają zwykle aktywność różnych pedantów, którzy usiłują koniecznie wymyślić tak skomplikowaną funkcję, żeby sieć neuropodobna w żaden sposób nie mogła się jej nauczyć. Z kolei zwolennicy "mocnej omnipotencjalności" sieci neuropodobnych pokazują wtedy koncepcje sieci o jakichś nieprawdopodobnie wielkich ilościach elementów, które jednak z dowolnie dużą dokładnością mogą takie złożone i dziwaczne funkcje odtwarzać. Trwa więc pewien wyścig absurdów – czy uda się sceptykom wymyślić tak dziwaczną funkcję, że jej neuronową aproksymacją nie da się w żaden sposób "ugryźć", czy też sportowo nastawieni twórcy rekordowo wielkich (i przez to niesłychanie wolno uczących się) nieliniowych sieci neuropodobnych – zdołają dla każdej z tych funkcji wymyślić jej neuronową realizację.

Wszystko to coraz bardziej oddala się od rzeczywistych problemów i potrzeb wynikających z praktyki i z tego powodu trzeba oceniać, że ta eskalacja absurdów nie służy postępowi zastosowań sieci. Wydaje się przeto, że celowe jest sformułowanie na gruncie techniki sieci neuropodobnych mniej wyzywającej, a przez to praktyczniejszej tezy, która brzmi: *można z dużym poziomem wiarygodności twierdzić, że sieci neuropodobne są w stanie zadowalająco modelować każdą dostatecznie gładką i regularną funkcję.*

Teza ta jest z punktu widzenia prowadzonych tu rozważań całkowicie wystarczająca, gdyż zdecydowana większość rzeczywistych zjawisk i procesów (fizycznych, biologicznych, gospodarczych itp.) **ma** wystarczająco regularny przebieg, żeby z powodzeniem mieścić się w obrębie wyżej zdefiniowanej klasy funkcji dostatecznie gładkich. Nawet

lokalne nieciągłości ("katastrofy") występujące w opisie niektórych zjawisk nie są przy tym przeszkodą nie do przebycia dla neuropodobnych modeli, zawodzą one dopiero wtedy, gdy ktoś usiłuje zdefiniować funkcję złożoną z ogromnej liczby szybko po sobie następujących katastrof – czyli w sytuacji, gdy to, co winno być wyjątkiem, staje się regułą. Wyjąwszy jednak te patologicznie zwyrodniałe problemy można przyjąć, że metoda rozwiązywania różnych praktycznych zadań, odwołująca się do tworzenia dla nich modeli neuronowych, jest bardzo skuteczna i godna polecenia. W szczególności użyteczne są sieci wielowarstwowe mogące tworzyć modele nieliniowe, gdyż ich uczenie pozwala często uzyskiwać użyteczne informacje bez konieczności drążenia związków typu przyczynowego. Badania i rozwój tego typu modeli mogą być źródłem wielu ciekawych osiągnięć naukowych i praktycznych w ciągu przynajmniej kilku najbliższych lat.

Literatura

Eremin D.M. (2001) A control system based on the neural networks technology. *Pribori i Sistemi Upravlenie, Kontrol, Diagnostika.* **9**, 8-11.

Horn J. (2001) Trajectory tracking of a batch polymerization reactor based on input-output-linearization of a neural process model. *Computers & Chemical Engineering*. **25**, 11-12, 1561-1567.

Jagannathan S., Tohmaz A. (2001) Congestion control of ATM networks using a learning methodology. *Proceedings of the 2001 IEEE International Conference on Control Applications,* 135-140.

Jih-Gau Juang and Kai-Chung Cheng (2001) Wind disturbances encountered during controlled landings using neural network approaches. *Proceedings of the 2001 IEEE International Conference on Control Applications*. 835-840.

Padhi R. and Balakrishnan, S.N. (2001) An optimal control based treatment strategy for parturient paresis using neural networks. *Proceedings of the 2001 IEEE International Conference on Control Applications.* 564-569.

Salcic Z. (2001) GSM mobile station location using reference stations and artificial neural networks. *Wireless Personal Communications.* **19**, 3, 205-226.

Shao Qing, Feng Ru-peng, Tong Wei-ming (2001) New algorithm for identification of fuzzy CP network based nonlinear dynamic systems. *Journal of the Harbin Institute of Technology.* **33**, 5, 721-724

Toyoda Y., Nakano K., Matsuo T., Higuchi K. (2001) Neuro-based modularized modeling and its application to deaerator with level control systems. In: *12th IFAC Symposium on System Identification.* **2**, 587-592.

Wei-Song Lin, Chun-Sheng Chen (2001) Robust neurofuzzy controller design of a class of uncertain multivariable nonlinear systems. *Proceedings of the 2001 IEEE International Conference on Control Applications,* 902-907.

Wen-Jing Li, Lee T. (2001) Hopfield neural networks for affine invariant matching. *IEEE Transactions on Neural Networks.* **12**, 6, 1400-1410.

ISBN 83-85847-78-2