Kernel. — 2004 nr 1 s. 16–19

Sztuczne Sieci Neuronowe

Dr Zdzisław Stęgowski

Wydział Fizyki i Techniki Jądrowej, AGH

Wstęp

W artykule tym chciałbym przybliżyć czytelnikowi pojęcie „Sztuczne Sieci Neuronowe" (w skrócie SSN). Pojęcie to nie powinno być obce osobom interesującym się lub studiującym informatykę. Poznając bliżej SSN będziemy dysponować metodę do rozwiązania zagadnień, z którymi inne metody nie radzą sobie. Ponadto SSN należą do grupy pojęć z zakresu Sztucznej Inteligencji, co w stosunku do pogody a zwłaszcza polityki daje znacznie ciekawszy temat do rozmowy towarzyskiej.

Realizacja SSN na klasycznym komputerze to wygenerowanie kilku macierzy o określonych wymiarach a następnie iteracyjna zmiana wartości elementów tych macierzy według określonego algorytmu i zadanej funkcji celu. Korzystając z gotowych pakietów do projektowania i uczenia SSN wystarczy napisać kilka linijek programu, aby zaprojektować i nauczyć taką sieć. Wygląda to tak prosto, że nie pozostaje nic innego jak zobaczyć, co to są te SSN i jak je można praktycznie realizować. Przed przystąpieniem do opisu SSN pozwolę sobie na jeszcze dwa akapity w ramach wstępu.

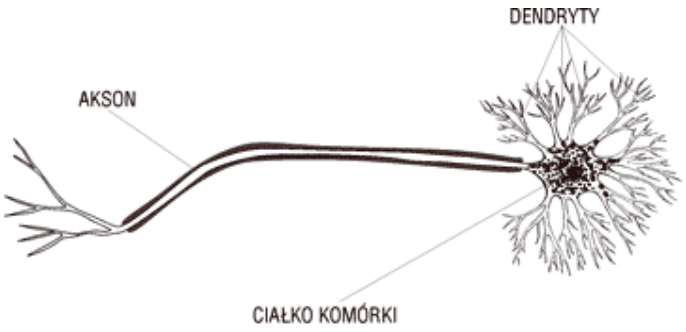
Z dużą satysfakcją należy podkreślić fakt, że pionierem zastosowań i promocji SSN w Polsce był i jest Prof. Ryszard Tadeusiewicz, obecny Rektor naszej uczelni. W roku 1993 wydał pierwszą książkę w Polsce na temat SSN pt. „Sieci neuronowe" a w roku 1999 ukazała się jego książka pt. „Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami".

Często stosowanym sposobem ukazania popularności i powszechności zastosowań SSN jest podawanie listy dziedzin nauki i techniki gdzie są one stosowane. Obecnie zastosowania te są tak szerokie, że lista taka byłaby nudną lekturą dla czytelnika. Dla ukazania powszechności zastosowań SSN można wykorzystać prawo popytu i podaży. O tym jak duży jest popyt na oprogramowanie SSN świadczy fakt, że większość znaczących firm produkujących programy obliczeniowe oferuje wydzielone pakiety do projektowania SSN i ich stosowania. Z powszechnie znanych programów obliczeniowych można tutaj wymienić pakiet STATISTICA i MATLAB. Innym bezspornym dowodem efektywności zastosowań

SSN jest fakt zatrudniania w znaczących firmach zbrojeniowych, co najmniej kilkudziesięcioosobowych zespołów pracujących w tematyce sztucznej inteligencji.

Mózg jako pierwowzór SSN

Każdy opis działania SSN zaczyna się od opisu układu nerwowego człowieka a zwłaszcza jego mózgu, oczywiście w stopniu, w jaki te zagadnienia są nam znane w chwili obecnej. Mózg składa się z komórek nerwowych, których liczba wynosi około stu miliardów (10^{11}). Budowa pojedynczej komórki przedstawiona jest na Rys. 1.



Rys. 1. Przybliżony wygląd komórki nerwowej

W mózgu komórki te poprzez akson, dendryty i synapsy połączone są ze sobą tworząc rozbudowaną sieć. O jej złożoności może świadczyć fakt, że niektóre komórki posiadają do tysiąca połączeń z innymi komórkami. Zadaniem takiej komórki, oprócz podtrzymywania czynności życiowych, jest odbieranie, przetwarzanie i przekazywanie informacji oraz jej zapamiętywanie. Ponieważ na sposób przetworzenia informacji wpływa stan pamięci a stan ten zależy od wcześniejszych informacji, działanie mózgu ma charakter dynamiczny. Prostym przykładem tej dynamiki, jest nasze zachowanie i sposób rozmowy z osobą, którą spotykamy po raz pierwszy (brak jakichkolwiek informacji w pamięci na temat tej osoby) i już odmienne zachowanie przy kolejnych spotkaniach, kiedy posiadamy coraz więcej informacji w pamięci o tej osobie. Nie wnikając w szczegóły, procesy informatyczne w sieci neuronowej mózgu mają charakter procesów elektro-chemiczno-biologicznych. Jedną z istotnych cech przekazywania informacji pomiędzy neuronami jest stosunkowo długi czas trwania takiego procesu, który jest rzędu milisekund. Pomimo tego nasz mózg działa często znacznie szybciej od klasycznego komputera, gdzie czasy przetwarzania informacji są o kilka rzędów wielkości

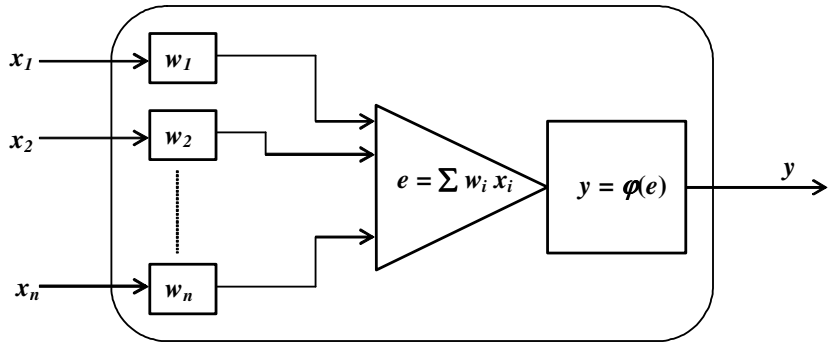
krótsze. Przykładem tego jest rozpoznawanie osoby ze zdjęcia. Ta umiejętność jest wynikiem równoległego charakteru przetwarzania informacji w sieci neuronowej oraz analizowaniu nie wszystkich informacji występujących w obrazie, lecz jego istotnych, wyróżniających go cech. Drugą istotną cechą sieci neuronowej mózgu jest rozproszenie pamięci danej informacji. W klasycznym komputerze dana informacja pamiętana jest w ściśle określonym miejscu, co przy fizycznym uszkodzeniu tego miejsca powoduje całkowitą jej utratę. W mózgu dana informacja jest rozproszona w różnych komórkach, dzięki czemu uszkodzenie komórki a nawet pewnej grupy komórek nie powoduje utraty informacji. Ta cecha była prawdopodobnie głównym powodem, że natura wybrała dla nas taką budowę i drogę rozwoju naszego mózgu. Zagadnienie pamięci nie jest jeszcze do końca rozpoznane, ale o stanie pamięci decydują również połączenia pomiędzy komórkami. Komórki mózgu oraz ich podstawowa struktura połączeń powstaje w okresie płodowym. W okresie naszego życia komórki zaczynają obumierać. Niedawno odkryto, że w ciągu całego czasu życia tworzone są nowe połączenia pomiędzy komórkami, czyli struktura sieci posiada również charakter dynamiczny. Należy tutaj zaznaczyć, że te dynamiczne zmiany zależą w dużym stopniu od intensywności używania naszego mózgu. Oznacza to możliwość pozostania w doskonalej formie intelektualnej do końca naszego fizycznego życia, jeżeli intensywnie będziemy się posługiwać naszym mózgiem w ciągu całego czasu, jaki jest nam dany do przeżycia.

Aby poznać i opisać działanie naszego mózgu należy odpowiedzieć na pytanie, czy złożoność działania naszego umysłu wynika ze złożoności procesów zapamiętywania i przetwarzania informacji zachodzących w komórce nerwowej, czy te procesy od strony informatycznej są stosunkowo proste natomiast cała finezja działania mózgu zawarta jest w liczebności sieci i strukturze jej połączeń? Próba odpowiedzi na to pytanie i ewentualna możliwość pełnego opisu działania naszego mózgu generuje znacznie szersze pytanie natury filozoficznej. Czy układ (np. człowiek) jest w stanie sam siebie dokładnie poznać i opisać? Prywatnie mam nadzieję, że odpowiedź na to pytanie brzmi NIE.

Model komórki nerwowej

Na wstępie należy zaznaczyć, że celem sztucznych sieci neuronowych jest zamodelowanie układów bazujących na strukturze naturalnych sieci neuronowych w celu rozwiązywania problemów, w których inne modele i algorytmy nie dają zadawalających wyników. Celem SSN nie jest badanie zachowania naturalnych sieci nerwowych, natomiast istnieje oczywiście sprzężenie zwrotne pozwalające na przenoszenie pewnych obserwowanych zachowań SSN na zachowanie naturalnych sieci.

Model sztucznego neuronu przedstawiony jest na Rys. 2. Na neuron taki podajemy wektor wejściowy ***X*** (czyli wartości x_1 ..... x_n) i po jego przetworzeniu otrzymujemy na wyjściu wartość *y*. Podstawowymi elementami neuronu są wartości wag *w*, funkcja wewnętrznego przetwarzania i funkcja aktywacji. Wartość sygnału wyjściowego z neuronu obliczana jest w dwu etapach.



Rys. 2. Model neuronu

W pierwszym etapie sygnały wejściowe *x* przemnażane są przez odpowiadające im wagi *w* i poddawane zadanej funkcji. Etap ten nazywany jest funkcją wewnętrznego przetwarzania i w praktyce jest to najczęściej funkcja sumowania (iloczyn skalarny wektora ***X*** i ***W***), natomiast można tam realizować dowolne inne funkcje takie jak: iloczyn, maksimum, minimum itp. W drugim etapie wynik funkcji wewnętrznego przetwarzania *e* podlega działaniu określonej funkcji wejścia-wyjścia zwanej w tym przypadku funkcją aktywacji φ. Ostatecznie dla pojedynczego neuronu sygnał wyjściowy *y* wyliczany jest z następującej zależności:

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i\right) = \varphi(W \bullet X) \tag{1}$$

gdzie:

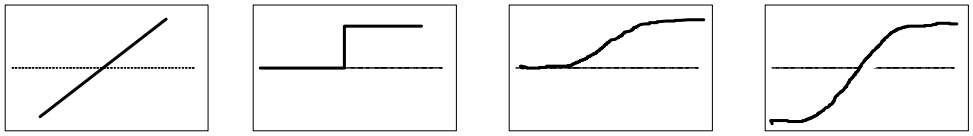
X - wektor danych wejściowych

W - wektor wag

φ - funkcja aktywacji

y - sygnał wyjściowy

W praktyce stosowane są funkcje aktywacji przedstawione na Rys.3.

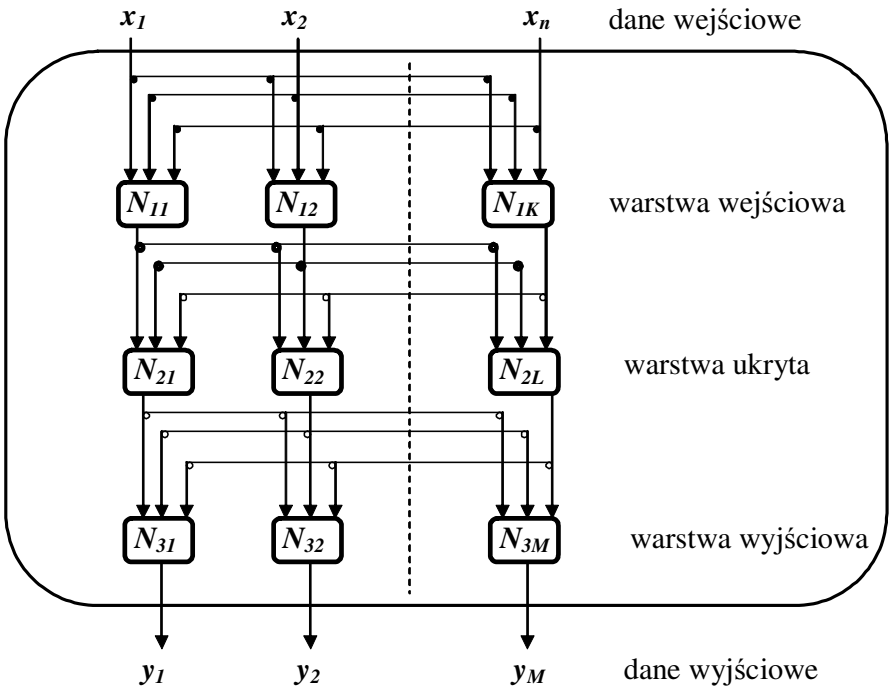


funkcja liniowa | funkcja skoku jednostkowego | funkcja sigmoidalna | funkcja tangens hiperboliczny

Rys.3. Cztery podstawowe rodzaje funkcji aktywacji stosowane w SSN

Sztuczne Sieci Neuronowe

Ze względu na topologię najczęściej w praktyce stosowane są sieci wielowarstwowe. W takiej sieci rozróżniamy warstwę wejściową, warstwy ukryte i warstwę wyjściową (Rys.4).



Rys. 4. Sieć trójwarstwowa o połączeniach zupełnych z jednokierunkowym przepływem sygnału

Połączenia między warstwami mogą mieć różną strukturę, lecz zazwyczj są to połączenia zupełne, co oznacza, że każdy neuron danej warstwy połączony jest ze wszystkimi

neuronami warstwy po niej następującej. W tego typu sieci przepływ sygnału ma charakter jednokierunkowy. Matematycznie funkcję realizowaną przez taką sieć możemy zapisać:

$$Y = \varphi_{wy}\left\{W_{wy} \bullet \varphi_{ukr}\left[W_{ukr} \bullet \varphi_{we}\left(W_{we} \bullet X\right)\right]\right\} \tag{2}$$

gdzie:

$\boldsymbol{X}$ - wektor danych wejściowych

$\boldsymbol{W}$ - macierz wag dla odpowiedniej warstwy

φ - funkcja aktywacji dla odpowiedniej warstwy

$\boldsymbol{Y}$ - wektor danych wyjściowych

Celem działania takiej sieci jest realizacja określonej funkcji, czyli uzyskanie żądanej odpowiedzi przy zadanych wielkościach wejściowych. Cel ten uzyskuje się poprzez proces uczenia sieci, czyli odpowiednią zmianę wartości wag neuronów. Jedną z podstawowych metod uczenia jest tak zwana *metoda z nauczycielem* (*uczenie nadzorowane*). W metodzie tej algorytm uczenia polega na przedstawieniu sieci zbioru uczącego, składającego się z danych wejściowych $\boldsymbol{X}$ i odpowiadającego mu danych wyjściowych $\boldsymbol{Z}$. Zbiór danych wejściowych przetwarzany jest przez sieć a uzyskany wynik $\boldsymbol{Y}$ porównywany jest z posiadanymi danymi wyjściowymi $\boldsymbol{Z}$. Różnica pomiędzy wartościami $\boldsymbol{Y}$ i $\boldsymbol{Z}$ stanowi podstawowy parametr do zmian wartości wag neuronów tak, aby osiągnąć minimum funkcji kryterialnej, którą standardowo stanowi suma kwadratów różnic pomiędzy wartościami $\boldsymbol{Y}$ i $\boldsymbol{Z}$. Podstawową metodą minimalizacji funkcji kryterialnej jest gradientowa metoda największego spadku, z której otrzymujemy zależność na zmianę wartości wag w kolejnych krokach iteracji zwaną *regułą delta.* Regułę tą można bezpośrednio stosować dla sieci jednowarstwowych. W przypadku sieci wielowarstwowych, kiedy sygnał z pierwszej warstwy nie jest sygnałem końcowym, konieczne było wprowadzenie dodatkowego algorytmu pozwalającego na równoczesną zmianę wag neuronów we wszystkich warstwach. Algorytm taki ze względu na jego działanie, czyli *rzutowanie* błędu z danej warstwy na warstwę poprzedzającą, został nazwany metodą *wstecznej propagacji błędu*. Metoda ta została opublikowana w roku 1985 i dała początek bardzo dynamicznemu rozwojowi prac związanych ze SSN oraz ich zastosowaniem.

W przypadku SSN proces programowania to nie tylko etap konstruowania sieci, lecz w głównej mierze etap uczenia i ewentualnie kolejne etapy douczania sieci. Istnieje tutaj piękna i nieprzypadkowa analogia do naszego życia. Do czasu, kiedy podejmiemy intelektualnie istotną pracę zawodową musimy przejść przez kolejne etapy edukacji, czyli:

przedszkole, szkołę podstawową, gimnazjum, liceum, studia a potem w pracy Bóg wie ile szkoleń i kursów.

Metoda *wstecznej propagacji błędu* była pierwszą metodą pozwalającą na uczenie sieci wielowarstwowych. W przypadku sieci liczebnie rozbudowanych metoda ta jest stosunkowo wolna. Pomimo wprowadzenia do tej metody elementów przyspieszających proces uczenia, takich jak metoda *momentum* czy adaptacyjną zmianę współczynnika uczącego, nadal poszukiwano i poszukuje się szybszych algorytmów uczących. Przykładowo w nowym pakiecie MATLABa do SSN pojawiły się nowe szybkie algorytmy uczące takie jak: *wsteczna propagacja Levenberga-Marquardta*, *sprzężona wsteczna propagacja Powella-Beala* i jeszcze kilka innych. Przepraszam, że używam tutaj pojęć, które nie są odpowiednio wyjaśnione, ale w tym miejscu chciałem pokazać jak dynamicznie prowadzone są prace nad rozwojem SSN.

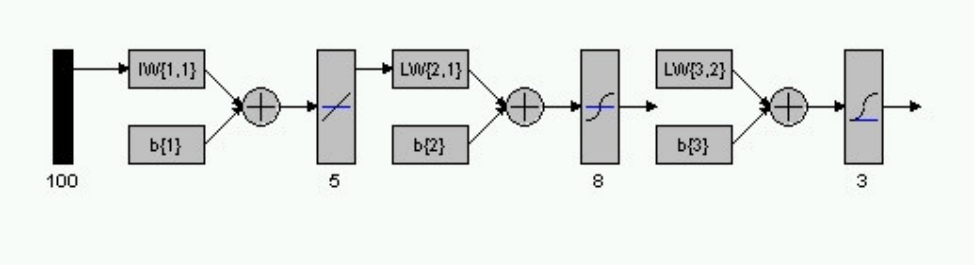
Realizacja SSN w MATLAB-ie

Korzystając z gotowych pakietów to w chwili obecnej projektowanie i uczenie SSN ogranicza się to do napisanie kilku linijek programu. Pokażę to na przykładzie pakietu *Neural Network Toolbox* pracującego w środowisku MATLABa. Wybór ten wynika z dwu powodów. Po pierwsze znam ten pakiet i używam go a po drugie jest on dostępny, poprzez UCI, dla wszystkich pracowników i studentów AGH. Oprogramowanie MATLAB (skrót od MATrix LABoratory) wykonane zostało na bazie języka C i posiada dużo użytecznych funkcji ułatwiających operację na macierzach. Hasłem przewodnim MATLABa jest: *po co tracić życie na pisanie pętli* i jako użytkownik mogę stwierdzić, że jest to prawda a nie slogan reklamowy (wcześniej pracowałem w FORTRANIE). Z doświadczenia mojej pracy ze studentami mogę stwierdzić, że osoby mające wcześniej styczność z językiem typu C lub FORTRAN, po pierwszych dwugodzinnych zajęciach mogą praktycznie samodzielnie rozpocząć prace w środowisku MATLABa.

Wracając do SSN to, jeżeli mamy przygotowane zbiory danych uczących $\boldsymbol{X}$ i $\boldsymbol{Z}$ (patrz poprzedni rozdział) i chcemy zaprojektować określoną sieć to wystarczy napisać następującą linijkę programu:

```
net = newff (minmax(X) , [5 8 3] , { 'pulelin' 'tansig' 'logsig' }, 'traingda');
```

Realizacja tej linii spowoduje, że pod nazwą *net* będziemy mieli sieć trójwarstwową o liczbie komórek, w kolejnych warstwach, równej 5, 8, 3 i odpowiednich funkcjach aktywacji: liniowa (*pulelin*), tangens hiperboliczny (*tansig*) i sigmoidalna (*logsig*) , oraz algorytm uczący wstecznej propagacji błędu oparty na metodzie gradientu z adaptacyjną zmianą współczynnika uczącego (*traingda*). Dla graficznego zobrazowania takiej sieci można poprzez narzędzie *nntool* wczytać taką sieć i aktywując instrukcję *view* otrzymać obraz przedstawiony na Rys. 5.



Rys. 5. Obraz zaprojektowanej sieci" net"

W sieci tej (*net*) znajduje się ponadto wiele innych parametrów (np. liczba kroków iteracji, rodzaj funkcji uczącej, minimalna wartość gradientu itd.), które możemy zdeklarować w kolejnych liniach programu a jeżeli tego nie zrobimy to przyjmowane są arbitralnie „narzucone" wartości. Kolejna linia programu wywołuje proces uczenia i ma ona postać:

net = train(net,X,Z);

Realizacja tej linii wywołuje proces uczenia, w którym zmieniane są wagi sieci *net*, zgodnie z zadanym algorytmem tak, aby zminimalizować błąd pomiędzy odpowiedzią sieci a zbiorem uczącym $\boldsymbol{Z}$. W czasie tego procesu wyświetlany jest wykres zmian wartości błędu w kolejnych krokach iteracji. Uzyskanie na tym etapie dużej zgodności (małego błędu) pomiędzy wartościami $\boldsymbol{Z}$ a wartościami uzyskanymi z sieci nie daje jeszcze podstaw do stwierdzenia, że sieć spełnia nasze wymagania. Na tym etapie może nastąpić tak zwane *przeuczenie sieci* lub *nauczenie na pamięć*. Chodzi o to, że jeżeli dla tak „dobrze" nauczonej sieci wprowadzimy dane, na których sieć nie była uczona to uzyskamy wyniki obarczone dużym błędem. Aby sprawdzić czy sieć nie jest obarczona w/w wadą należy posiadać w „zapasie" tak zwany zbiór danych testujących $\boldsymbol{X}_t$, $\boldsymbol{Z}_t$. Dane te nie mogą być użyte w procesie uczenia. W tym momencie dla nauczonej sieci *net* podajemy dane wejściowe $\boldsymbol{X}_t$ i wyliczamy odpowiedź sieci $\boldsymbol{Y}_t$. W naszym przypadku realizujemy to instrukcją:

$\boldsymbol{Y}_t = sim(net, \boldsymbol{X}_t);$

Teraz wystarczy porównać wartości $\boldsymbol{Z}_t$ z $\boldsymbol{Y}_t$. i podjąć decyzję czy uzyskujemy zadawalające nas wyniki czy nie. Powyższego przykład pokazuje, że posiadając odpowiedni program wystarczy napisać kilka linijek instrukcji i mamy zaprojektowaną, nauczoną i przetestowaną sieć neuronową. Podany przykład jest stosukowo prostym przypadkiem sieci z jednokierunkowym przepływem sygnału (informacji) i metodą uczenia z *nauczycielem*. Tego typu sieci są obecnie powszechnie stosowane. Z ustnych przekazów (czyli z nieznaną dokładnością) słyszałem, że jest to około 80% zastosowań SSN.

W tym miejscu nasuwają się następujące pytania:, jaką architekturę sieci wybrać na wstępie, co robić, jeżeli wyniki uczenia sieci nas nie zadawalają, czy dane używane do uczenia sieci należy wstępnie przetwarzać a jeżeli tak to jak to robić, jak projektować sieci ze sprzężeniami zwrotnymi, jak podawać sygnały wejściowe z opóźnieniem itd. Na wszystkie te pytania istnieją mniej lub bardziej precyzyjne odpowiedzi, ale o tym już może przy następnej okazji.

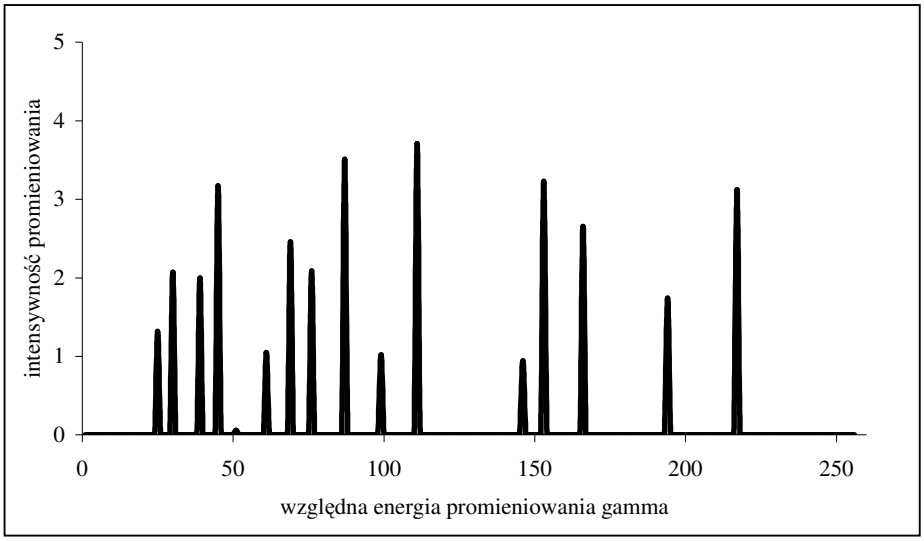
Przykład zastosowania SSN

Na zakończenie przedstawię przykład, z własnego podwórka, zastosowania SSN. Jednym z szerszych zastosowań SSN jest rozpoznawanie, klasyfikacja czy odszumianie obrazów. Pojęcie obrazu ma w tym przypadku znacznie szersze znaczenie od potocznego rozumienia tego słowa. Jako obraz możemy przykładowo rozumieć funkcję zależności uzyskanych na drodze pomiaru. Przykładem takiej funkcji jest tak zwane widmo spektrometryczne uzyskane z pomiaru próbki zawierającej izotopy promieniotwórcze emitujące promieniowanie gamma o różnych energiach. Aby nie zagłębiać się tutaj w zagadnienia fizyki jądrowej przedstawię to następująco. Załóżmy, że mamy próbkę złożoną z izotopów emitujących fotony gamma o 16 różnych energiach. Równocześnie intensywność emisji poszczególnych fotonów (liczba emitowanych fotonów na jednostkę czasu) jest różna dla różnych energii.

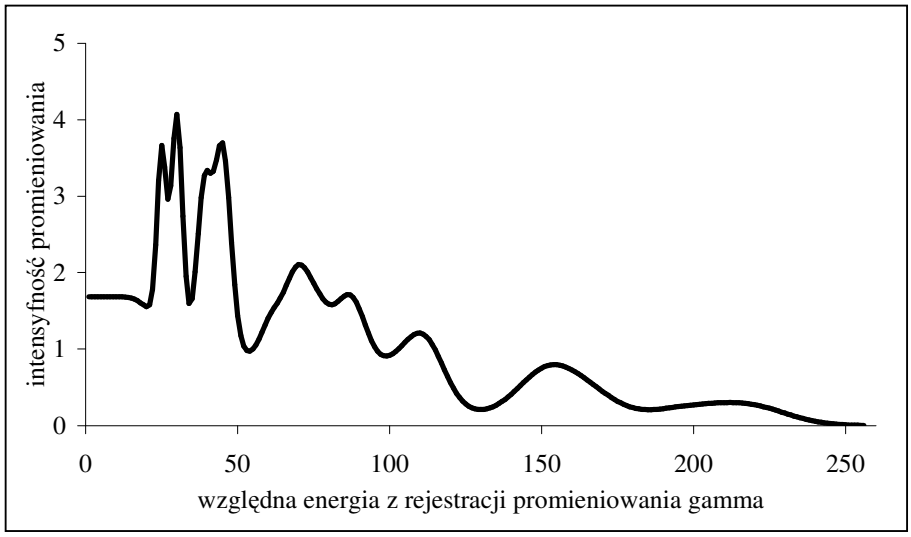
Zakładając hipotetycznie, że posiadamy idealny detektor, spektrometryczne widmo z takiej próbki wyglądałoby tak jak na Rys. 6 i można je nazwać widmem pierwotnym. Na osi poziomej tego wykresu jest energia a oś pionowa odpowiada intensywności promieniowania. Mając taki wykres bez problemu odczytujemy energię fotonów oraz ich intensywność

promieniowania, czyli wielkości, o które nam chodzi w analizie spektrometrycznej. Problem w tym, że taki detektor nie istnieje.

Jednym z powszechnie stosowanych detektorów promieniowania gamma jest detektor scyntylacyjny. Należy on do grupy detektorów spektrometrycznych, lecz ze względu na określone procesy detekcji i wzmacniania sygnału powoduje on stosunkowo duże zniekształcenia widma spektrometrycznego. Na Rys. 7 pokazane jest widmo pomiarowe z takiego detektora, którego odpowiednikiem jest widmo pierwotne (Rys. 6). Analiza spektrometryczna takiego widma pomiarowego jest trudna a często niemożliwa do



Rys. 6. Widmo pierwotne promieniowania gamma



Rys. 7. Widmo promieniowania gamma z detektora scyntylacyjnego

wykonania. Na wskutek rozmycia pików, pochodzących od różnych energii fotonów gamma, następuje ich nakładanie i są one w widmie nierozróżnialne. Z tych samych przyczyn wyznaczenie intensywności promieniowania dla poszczególnych energii fotonów gamma jest często niewykonalne. Rozwiązaniem byłoby przekształcenie widma pomiarowego w widmo pierwotne, tylko jak to zrobić?

Potraktujmy proces detekcji detektorem scyntylacyjnym jako transformację widma pierwotnego w widmo pomiarowe. Jeżeli ta transformacja jest jednoznaczna to wystarczy zrobić transformację odwrotną i problem mamy rozwiązany. Łatwo napisać, ale trudno wykonać. Podstawowy problem tkwi w matematycznym określeniu funkcji transformacji a wynika to z braku precyzyjnego modelu opisu detekcji i wzmacniania sygnału w detektorze scyntylacyjnym.

W przypadku braku odpowiednich modeli podstawową metodą działania jest zebranie odpowiedniej liczby danych doświadczalnych i określenie szukanych zależności. Są to najczęściej analityczne zależności dla funkcji wynikających z ogólnych teorii, ale o nieznanych parametrach, lub arbitralnie ustalonych funkcji wynikających z analizy danych doświadczalnych i aproksymacji tych funkcji do tych danych. W nauce a zwłaszcza w technice metoda ta jest szeroko stosowana. W przypadku arbitralnie ustalonych funkcji na postawie obrazu (wykresu), jaki uzyskujemy z danych doświadczalnych metoda ta daje dobre wyniki dla jedno lub dwu wymiarowych funkcji. Tutaj nasza wyobraźnia, wytrenowana przez nauczycieli matematyki, pozwala nam tworzyć obrazy funkcji jednowymiarowych bez większych problemów. Z funkcją dwuwymiarową już zaczyna być trudniej, choć w dniu dzisiejszym programy graficzne pozwalają na łatwe obrazowanie takich zależności i oglądanie ich pod dowolnymi kątami (w dosłownym znaczeniu). Sprawa staje się bardzo trudna, kiedy nie tylko dziedzina funkcji ma charakter wielowymiarowy, ale i wartości funkcji są wielowymiarowe. Do takich przypadków doskonale nadają się SSN i zagadnienie to nazywane jest aproksymacją wielowymiarową.

Wracając do naszego przykładu okazuje się, że wystarczy bardzo prosta jednowarstwowa liniowa SSN pozwalająca na przekształcenie pomiarowego widma spektrometrycznego (Rys. 7) w widmo pierwotne (Rys. 6). Oczywiście, aby otrzymać taką sieć należy przygotować odpowiedni zbiór uczący i przeprowadzić proces uczenia tej sieci. Łącząc taką SSN z detektorem scyntylacyjnym możemy otrzymać „idealny detektor", który pozwala uzyskać widmo pierwotne. Jak zawsze tak i tutaj są pewne ograniczenia.

Podstawowym ograniczeniem jest to, że taki układ nie rozpozna energii fotonu gamma, jeżeli nie była ona uwzględniona w zbiorze uczącym. Na dokładkę wystąpienie takiej energii zafałszuje wyniki intensywności promieniowania fotonów o energii zbliżonej do tej nieznanej przez sieć energii. W przypadku, kiedy SSN była uczona dla danej energii fotonu a w widmie pomiarowym ona nie występuje, to sieć prawidłowo podaje, że intensywność promieniowania takich fotonów wynosi zero. I znowu mamy piękną analogię do działania naszego mózgu, czyli nie rozpoznamy osoby na zdjęciu, jeżeli jej nie znamy (trywialne), ale jeżeli jest tam osoba bardzo podobna do kogoś, kogo znamy to możemy fałszywie ją rozpoznać. A nie daj Boże to zdjęcie pokazuje nam prokurator i konsekwencje dla fałszywie rozpoznanej osoby mogą być bardzo nieprzyjemne. Natomiast, jeżeli kogoś dobrze znamy to łatwo możemy stwierdzić, że go nie ma na tym zdjęciu.

Przedstawiony problem należy do dziedziny aproksymacji wielowymiarowej, ale jeżeli ktoś się pogubił w tej wielowymiarowości to może to potraktować jako odszumianie, czyli wydobywanie istotnych cech z zaszumionego obrazu. Z zagadnieniami tego typu lub bardzo podobnymi mamy, do czynienie w powszechnie obecnie stosowanej tomografii komputerowej. Niestety trudno jest znaleźć szczegółowe publikacje na ten temat. Wynika to z faktu, że jest to z zakresu *know-how*, czyli coś, za co trzeba płacić. I tak jest w wielu innych technicznych zastosowaniach SSN.

Podsumowanie

Mam nadzieję, że artykuł ten przybliżył czytelnikowi pojęcie „Sztuczne Sieci Neuronowe”, pokazując ich genezę oraz podstawową strukturę i przykład zastosowania. Dostępna obecnie literatura i oprogramowanie pozwala na podjęcie samodzielnych prób „zabawy” ze SSN. W takim przypadku ważne jest, aby mieć ściśle określony problem i cel, jaki chce się osiągnąć. Osobom mniej samodzielnym, a zainteresowanym SSN, polecam kursy na temat SSN. Ze swojej praktyki uważam, że istotnym elementem takich kursów jest możliwość samodzielnego wykonania określonych projektów. Samo wysłuchanie lub przeczytanie, czasami bardzo ekscytujących opowieści o Sztucznych Sieciach Neuronowych czy Sztucznej Inteligencji to za mało, aby potem coś samodzielnie zrobić.

Książki na temat SSN

- Duch W., Korbacz J., Rutkowski L., Tadeusiewicz R., *Sieci Neuronowe* (Biocybernetyka I Inżynieria Biomedyczna 2000 Tom 6.) Akademicka Oficyna Wydawnicza EXIT

- Korbicz J., Obuchowicz A., Uciski D. *Sztuczne Sieci Neuronowe Podstawy i Zastosowania*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994.

- Osowski S.: *Sieci neuronowe w ujęciu algorytmicznym*, WNT, Warszawa 1996.

- Osowski S.: *Sieci neuronowe do przetwarzania informacji*, Oficyna Wydawnicza PW, Warszawa 2000.

- Tadeusiewicz R.: *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa 1993

- Tadeusiewicz R.: *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999.

- Żurada J., Barski M., Jędruch W., "*Sztuczne sieci neuronowe*", Wydawnictwo Naukowe PWN, Warszawa 1996.

Strony WWW na temat SSN

- http://www.ftj.agh.edu.pl/~STEGOWSKI/sn.htm
- http://republika.pl/edward_ch/
- http://irm.wsm.szczecin.pl/wwwzirm/SN1.htm
- http://www.mathworks.com/products/neuralnet/