Państwowa Uczelnia im. Stefana Batorego

Kolegium Medyczno-Przyrodniczo-Techniczne

Instytut Nauk Informatyczno-Technicznych

Kierunek: **Informatyka**

Specjalność: **Nazwa specjalności**

Bohdan Macukow

Nr indeksu: **XXXX**

Praca dyplomowa nt.:

SIECI NEURONOWE W RÓŻNYCH DZIEDZINACH NAUKI

Promotor:

stopień/tytuł naukowy imię i nazwisko

Pracę przyjmuję

.....
(data i podpis)

Skierniewice, 20XX

Streszczenie

Streszczenie pracy może zajmować co najwyżej jedną stronę i powinno zawierać (w co najwyżej trzech akapitach) następujące informacje: co było celem pracy i jakich zagadnień dotyczy praca, jaki problem został rozwiązany w pracy oraz jak to zostało zrobione i jakie są uzyskane wyniki.

Summary

Streszczenie w języku angielskim musi określać cel pracy, problem, jaki został rozwiązany oraz sposób jego rozwiązania. **Streszczenie musi być napisane poprawnym językiem angielskim.**

Spis treści

Lista zastosowanych skrótów

DBS – (ang. *Database Server*) serwer bazy danych

GUI – (ang. *Graphic User Interface*) graficzny interfejs użytkownika

Lista skrótów (akronimów), jaką Dyplomant zastosował w pracy jest obowiązkowa. Proponuje się Listę skrótów występujących w pracy utworzyć w porządku alfabetycznym.

Niezależnie od Listy używanych skrótów, każdy skrót użyty w pracy musi być wyjaśniony w miejscu jego pierwszego wystąpienia w tekście pracy. W dalszej treści pracy, można już posługiwać się tym skrótem, bez podawania jego rozwinięcia.

Przykładowe rozszerzenie nazwy obcojęzycznej powinno być wykonane w następujący sposób: Treść zdania....w kontekście kompatybilności elektromagnetycznej EMC (ang. *Electromagnetic Compatibility*).

1. Wstęp

Praca ta jest poświęcona sztucznym sieciom neuronowym – ciekawym i wielce obiecującym systemom przetwarzania informacji. Zainteresowanie sieciami wynika m.in. z faktu, że dostosowanie sieci do wykonania określonego zadania odbywa się zazwyczaj poprzez uczenie jej, przez użycie zestawu pobudzeń i odpowiadających im reakcji, a nie przez specjalne algorytmy, programy itp. Mówiąc o sieciach neuronowych często zamiennie używamy nazwy neurokomputery mając na myśli urządzenia, których budowa podobna jest do biologicznej struktury mózgu ludzkiego bądź która działa tak jak działałby mózg.

Dzisiejsze, tak szerokie i powszechne zainteresowanie sieciami neuronowymi zarówno wśród inżynierów, przedstawicieli nauk ścisłych - matematyki i fizyki oraz biologów czy neurofizjologów wynika przede wszystkim z poszukiwań nad sposobami budowy bardziej efektywnych i bardziej niezawodnych urządzeń do przetwarzania informacji a układ nerwowy jest tutaj niedościgłym wzorem. Mózg człowieka ciągle jest najpotężniejszym z istniejących obecnie urządzeń liczących do celów przetwarzania informacji w czasie rzeczywistym. Fascynacje mózgiem, jego własnościami (odpornością na uszkodzenia, równoległym przetwarzaniem itp.) już w latach 40-tych zaowocowały pracami, których fundamentalne znaczenie odczuwamy jeszcze dzisiaj.

Mózg i komputer, zastanówmy się jakie mamy tutaj podobieństwa i jakie różnice. Wyobraźmy sobie komputer, który rozwiązując pewien problem sam się uczy. Najpierw wprowadzamy do niego informacje o postawionym zadaniu, dane wejściowe problemu oraz wybrane przykłady wraz z poprawnymi ich rozwiązaniami. Następnie komputer analizuje wprowadzone informacje i ucząc się na swoich błędach osiąga w końcu taki stan, w którym postawiony problem może być rozwiązany. W takiej działalności można zauważyć wiele podobieństwa do działania człowieka. I tutaj pojawia się pytanie: czy potrafimy skonstruować urządzenie techniczne o podobnych właściwościach??? Badania ostatnich lat sugerują odpowiedź twierdzącą - a właśnie sieci neuronowe wydają się być drogą prowadzącą do tego celu.

Z punktu widzenia informatyki interesującym jest porównanie własności komputera z własnościami mózgu. Sieci nerwowe mogą przeprowadzać niezwykłe obliczenia i działania, aczkolwiek jest rzeczą oczywistą, że w na przykład obliczeniach arytmetycznych mózg nie jest tak dobrym urządzeniem (tak szybkim, wydajnym i dokładnym) jak komputer. Ale z drugiej strony. gdy ma się do czynienia z zadaniami takimi jak rozpoznawanie, skojarzenia czy klasyfikacja - mózg może pokonać nawet najszybszy superkomputer, pomimo że w tym procesie neurony jako jednostki przetwarzające są o wiele rzędów wielkości wolniejsze od swoich elektronicznych czy optoelektronicznych odpowiedników. Z punktu widzenia zasady działania, zarówno mózg, jak i konwencjonalne komputery realizują w zasadzie podobne funkcje - gromadzą, przetwarzają czy odzyskują informacje. Różnica nie leży więc w odmiennym działaniu lecz na odmiennych zasadach gromadzenia i przetwarzania informacji. Jeżeli mamy wykonać proste działanie arytmetyczne np. mnożenie dwu cyfr, to nie wykonujemy tego mnożenia lecz rozpoznajemy problem, a następnie przywołujemy z pamięci właściwą, skojarzoną z nim odpowiedź wynikającą z faktu, że kiedyś w dzieciństwie uczyliśmy

się tabliczki mnożenia. Jedną z najciekawszych własności, najbardziej różniącą świat sieci neuronowych od świata komputerów jest ich zdolność do tolerowania i poprawiania błędów. Mózg zdolny jest do rekonstrukcji, odtworzenia sygnału na podstawie informacji częściowej a dodatkowo obarczonej błędami.

Analizując metody przetwarzania i selekcji informacji oraz sposoby podejmowania decyzji w systemie nerwowym łatwo można dojść do wniosku, że jest on przykładem rozwiązania wielu problemów, z którymi od wielu lat boryka się informatyka, teoria przetwarzania informacji czy teoria optymalizacji.

Powróćmy jeszcze raz do genezy zainteresowania sieciami neuronowymi. Zafascynowani potęgą obliczeniową, ogromnymi szybkościami działania i możliwościami dzisiejszych komputerów często zapominamy czym naprawdę one są - jedynie doskonałymi liczydłami. Te znakomite urządzenia naprawdę są szalenie powolne i nieefektywne. Przecież do wykonania konkretnej operacji komputer wykorzystuje jedynie mikroskopijną część swoich ogromnych możliwości, używa jedynie kilku spośród swoich elementów - podczas gdy cała reszta pozostaje nieaktywna. przecież szeregowy system pracy wymusza na nim wykonywanie operacji w ustalonej kolejności. Oczywiście staramy się temu zaradzić. Tworzymy jednostki pracujące równolegle, używamy procesorów wektorowych zwielokrotniając możliwości obliczeniowe. Jednak wszystkie te usprawnienia nikną wobec potencjalnych możliwości sieci neuronowych, w których każdy, niezależnie pracujący neuron, jest jak gdyby niezależnym procesorem, a cała sieć zbudowana z tysięcy (lub milionów) takich procesorów może pracować w pełni równolegle i wykonywać wiele operacji równocześnie. Takie przetwarzanie pozwala sieciom niesłychanie efektywnie wykonywać złożone zadania (nawet zadania obliczeniowe), pomimo użycia bardzo powolnych elementów. Trzeba bowiem pamiętać, że typowy impuls neuronowy trwa kilka milisekund - czyli miliony razy dłużej niż w przypadku impulsów generowanych przez krzemowe układy półprzewodnikowe.

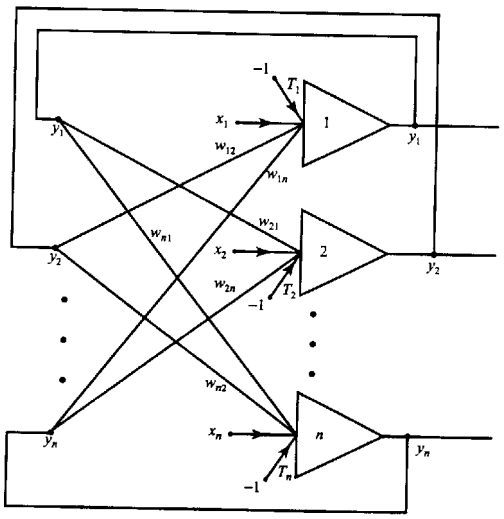
2. Możliwości zastosowań sieci neuronowych

Omówione cechy sieci neuronowych jak także znane z literatury różnorodne modele struktur sieciowych pozwalają na scharakteryzowanie ich możliwości oraz obszarów ich potencjalnych zastosowań. Sieci nie uczą się algorytmów, lecz uczą się przez przykłady. W przeciwieństwie do konwencjonalnych komputerów są słabymi maszynami matematycznymi i słabo nadają się do typowego przetwarzania opartego o algorytmy. Bardzo dobrze natomiast nadają się do zadań związanych z rozpoznawaniem obrazów (nawet tych o niepełnej bądź zafałszowanej informacji), do zadań optymalizacyjno - decyzyjnych, do szybkiego przeszukiwania dużych baz danych.

2.1. Sieci Hopfielda i Hamminga

Modele sieci Hopfielda i Hamminga są jednymi z najczęściej omawianymi, badanymi i wykorzystywanymi. Zazwyczaj obie są stosowane do rozpoznawania lub klasyfikacji obrazów, które są reprezentowane w sposób binarny. Warto także zaznaczyć, że sieć Hopfielda jest często podawana jako przykład pamięci skojarzeniowej lub jako układ do rozwiązywania zadań z zakresu optymalizacji.

Strulturę sieci Hopfielda można opisać bardzo prosto – jest to układ wielu identycvznych elementów połączonych metodą każdy z każdym. Jest zatem najczęściej rozpatrywana jako struktura jednowarstwowa. W odróżnieniu od sieci warstwowych typu perceptron sieć Hopfielda jest siecią rekurencyjną, gdzie neurony są wielokrotnie pobudzane w jednym cyklu rozpoznawania co uzyskuje się poprzez pętle sprzężenia zwrotnego.



Rys. 2.1 Jednowarstwowa sieć Hopfielda ze sprzężeniem zwrotnym.

Wagi połączeń wyliczane są w sieci Hopfielda a priori, jej faza uczenia ogranicza się do wyliczenia wartości wag zgodnie zasadą uczenia Hebba

$$w_{ij} = \begin{cases} \sum_{k=1}^{M} x_i^k x_j^k & \text{dla } i \neq j \\ 0 & \text{dla } i = j \end{cases}$$

Równanie 2.1

R

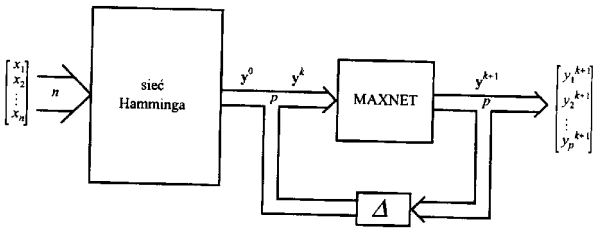
gdzie:

M jest ogólną liczbą zapamiętywanych wzorców

xi to i-ta składowa wzorca (górny indeks określa numer wzorca), xi -1,1}

W fazie odtwarzania na wejście sieci podany jest nieznany sygnał wejściowy i zadaniem sieci jest w procedurze rekurencyjnej „znaleźć" ten z zapisanych w jej strukturze wzorców do którego ten sygnał wejściowy jest najbardzie podobny.

Sieć Hamminga jest dwuwarstwowym klasyfikatorem o schemacie blokowym pokazanym na rys.4. Zadaniem sieci jest, podobnie jak w sieci Hopfielda, wyszukanie tego spośród zapamiętanych wzorców, który jak najbardziej podobny do sygnału wejściowego. Jako miarę podobieństwa przyjmuję się tzw. odległość Hamminga – czyli liczbę bitów rożnych w porównywanych obiektach. Tej selekcji dokonuje pierwsza warstwa klasyfikatora. Najsilniejszy sygnał wyjściowy neuronu jest wskaźnikiem najmniejszej odległości Hamminga pomiędzy sygnałem wejściowym a wzorcem klasy, którą ten neuron reprezentuje. Warstwa druga, zwana MAXNET odgrywa rolę pomocnicą. Jest to sieć rekurencyjna mająca za cel wytłumić sygnały wyjściowe wszystkich neuronów tej warstwy oprócz twego, który otrzymał na swoim wejściu najsilniejszy sygnał wejściowy.



Rysunek 2.2. Schemat blokowy klasyfikatora Hamminga

Wagi w pierwszej warstwie są ustalane podobnie jak w modelu Hopfielda metodą jednorazowego zapisu gdyż w praktyce są równe odpowiednim składowym zapisywanych w sieci wzorców.

Ponieważ odległość Hamminga pomiędzy wektorem wejściowym x oraz zapamiętanym sygnałem wzorcowym s(m) jest równa

$$HD(\mathbf{x},s^{(m)}) = \frac{1}{2}n - \frac{1}{2}xs^{(m)} \qquad (2)$$

Równanie 2.2

gdzie n jest liczbą bitów w rozpatrywanych wektorach, więc dodając na wejściu każdego elementu stały sygnał wejściowy n/2, otrzymujemy łączne pobudzenie elementu

$$net = \begin{bmatrix} n - HD(x,s^{(1)}) \\ .... \\ n - HD(x,s^{(p)}) \end{bmatrix} \qquad (3)$$

Równanie 2.3

W sieci MAXNET są pobudzenia pobudzające (autosprzężenie zwrotne z wagą 1) oraz hamujące, typu hamowanie oboczne, z wagą -ε, gdzie 0<ε<1/p (p- liczba elementów w każdej z warstw sieci).

3. Sieci neuronowe w zastosowaniu do rozwiązywania wybranych problemów optymalizacyjnych

Dzięki swojej budowie, dzięki zdolnościom do wykonywania obliczeń równoległych oraz wynikającym stąd możliwościom przetwarzania ogromnych ilości informacji sieci neuronowe bardzo dobrze nadają się do rozwiązywania złożonych, pracochłonnych i czasochłonnych problemów optymalizacyjnych. Szybkość przetwarzania w sieciach neuronowych stwarza ogromne możliwości przyspieszenia nawet bardzo złożonych obliczeń.

Podstawowe podejście do problemów optymalizacji polega na sprowadzeniu zadania do zagadnienia minimalizacji pewnej funkcji energetycznej opisującej pewną sieć rekurencyjną traktowaną jako swoisty układ minimalizujący. Inne stosowane podejście, to zaprojektowanie sieci neuronowej, w której neurony wzajemnie ze sobą rywalizują dążąc do przejścia ze stanu nieaktywnego w aktywny.

Bardzo dobrym przykładem sieci neuronowej do tego rodzaju zadań jest sieć Hopfielda. Jak wiadomo, działanie jej oparte jest na samorzutnym dążeniu sieci

do minimalizacji jej funkcji energii. Problem polega na odpowiednim przejściu od zadania minimalizacji funkcji celu postawionego problemu wyjściowego (z uwzględnieniem istniejących ograniczeń) do zagadnienia minimalizacji funkcji energetycznej sieci.

Wiąże się z tym konieczność rozwiązania następujących problemów:

- sposobu reprezentacji problemu przy użyciu sieci neuronowej, aby na podstawie jej stanu końcowego (wartości sygnałów wyjściowych elementów sieci) możliwe było określenie rozwiązania problemu wyjściowego,
- takiego określenie funkcji energetycznej, aby jej minimum odpowiadało optymalnemu rozwiązaniu problemu wyjściowego,
- określenia struktury, wag połączeń oraz wielkości pobudzeń zewnętrznych,
- określenia równań dynamiki poszczególnych elementów aby zapewnić zmniejszanie się wartości funkcji energetycznej,
- określenia wartości początkowych poszczególnych elementów.

W typowych problemach optymalizacji kombinatorycznej funkcję energetyczną najczęściej wybiera się w postaci

$$E = \sum_i A_i \,(\text{miara naruszenia i - tego ograniczenia}) + B*(\text{funkcja celu}) \qquad (4)$$

przy czym Ai i B są dodatnimi parametrami. Poprzez minimalizację funkcji energetycznej staramy się równocześnie zminimalizować wyjściową funkcję celu oraz zmaksymalizować stopień spełnienia ograniczeń.

Takim klasycznym i dobrze znanym problemem optymalizacji kombinatorycznej jest Problem Komiwojażera (Traveling Salesman Problem - TSP). Zadanie jest proste, komiwojażer ma objechać N miast. Planuje podróż w taki sposób aby każde miasto odwiedzić dokładnie jeden raz a następnie wrócić do punktu startu. Zadanie polega na minimalizacji długości przebytej trasy przy założeniu, że znamy odległości pomiędzy miastami. Problem ten posiada skończoną liczbę rozwiązań dopuszczalnych a mianowicie (N-1)!/2.

Problemy optymalizacji można podzielić na klasy według ich rozwiązywania Jeżeli istnieje algorytm, który ze wzrostem rozmiaru problemu rozwiązuje problem w czasie rosnącym tylko wielomianowo (lub wolniej), wtedy mówimy, że jest to problem wielomianowy i należy do klasy P. Klasa P jest podklasą klasy NP podobnie jak klasa tzw. problemów NP-zupełnych. Czas potrzebny do rozwiązania problemu NP.-zupełnego wzrasta wykładniczo ze wzrostem N. Ponieważ w przypadku Problemu Komiwojażera zastosowanie pełnego przeszukiwania przy większej liczbie miast nie jest możliwe (problem ten należy właśnie do klasy NP-zupełnych) stosowane są jedynie algorytmy przybliżone, które aczkolwiek nie są pozbawione wad jednak działają w czasie wielomianowym (będącym wielomianem zmiennej N). Podstawowym problemem jest określenie odpowiedniej reprezentacji danych.

W „sieciowym” rozwiązaniu każde miasto jest reprezentowane za pomocą jednego wiersza zawierającego N neuronów. W każdym wierszu dokładnie jeden neuron powinien

przyjmować wartość 1 a pozostałe wartość 0. Pozycja (od 1 do N), na której występuje neuron „z jedynką” odpowiada kolejności na trasie poruszania się komiwojażera.

Ogólna postać funkcji energetycznej ma postać

$$E = \frac{A}{2}\sum_X \sum_Y \sum_{i,i\neq j} v_{Xi}v_{Xj} + \frac{B}{2}\sum_i \sum_X \sum_{Y,Y\neq X} v_{Xi}v_{Yi} + \frac{C}{2}\left(\sum_X \sum_i v_{Xi} - N\right)^2 + \frac{D}{2}\sum_X \sum_{Y,Y\neq X} \sum_i d_{XY}v_{Xi}\left(v_{Y,i+1} - v_{Y,i-1}\right) \quad (5)$$

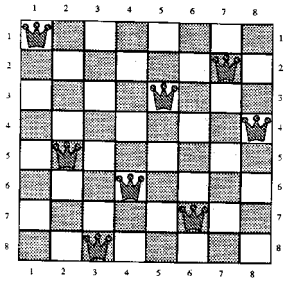
gdzie X, Y oznaczają miasta natomiast i,j - etapy, czyli vXi = 1 oznacza, że miasto X zostanie odwiedzone jako i-te z kolei.

Pierwszy składnik we wzorze (5) jest równy zero wtedy, gdy w każdym wierszu występuje tylko jedna jedynka - jest to więc rodzaj „kary” za wielokrotne odwiedzanie tych samych miast. Składnik drugi jest „karą” za równoczesny pobyt komiwojażera w dwóch różnych miejscach. Składnik trzeci równy jest zeru wtedy i tylko wtedy gdy dokładnie N neuronów jest w stanie pobudzonym (przeciwdziała zatem tendencjom minimalizacji dwóch pierwszych składników w taki sposób, aby żaden neuron nie był pobudzony). Te trzy składniki reprezentują ograniczenia problemu natomiast składnik czwarty odpowiada przebytej drodze. Jest on tak zbudowany, że sumowaniu podlegają tylko odległości dXY między miastami kolejno odwiedzanymi. Stałe A,B,C i D dobierane są heurystycznie.

Sieć składa się z N2 jednostek a liczba potrzebnych połączeń jest rzędu N3. Dobre rozwiązanie zależy w znacznym stopniu od właściwego doboru stałych A,B,C i D (5). Niestety algorytm jest stosunkowo wolno zbieżny a ponadto otrzymane rozwiązanie nie jest rozwiązaniem najlepszym lecz jest rozwiązaniem optymalnym.

Bardzo podobnym problemem (należącym do tej samej klasy) jest zagadnienie N - hetmanów polegające na ustawieniu na szachownicy N*N, o N2 polach, N hetmanów w taki sposób aby się wzajemnie nie atakowały. Liczba rozwiązań problemu dla kilku początkowych wartości N wynosi odpowiednio:

N = 1 1, N = 2 2, N = 4 2, N = 5 10, N = 8 92, N = 10 724 N = 12 14200 itd.



Przykładowe rozwiązanie dla N = 8 pokazane jest na rys. 3.1.

Rysunek 3.1 Przykładowe rozwiązanie problemu 8 hetmanów

Jako reprezentację komputerową zbioru neuronów tworzących sieć rozwiązującą ten problem (o wymiarze N), zastosowano macierz kwadratową VN*N. Zgodnie z ideą sieci Hopfielda strukturę oraz wagi poszczególnych połączeń pomiędzy neuronami należy dobrać w taki sposób, by minima funkcji energetycznej odpowiadały rozwiązaniom problemu tzn. spełniały zbiór ograniczeń problemu. Sieć startuje z punktu „odpowiednio bliskiego" rozwiązaniu, w trakcie ewolucji będzie zmniejszała swoją energię aż do osiągnięcia stanu odpowiadającego minimum funkcji energetycznej. Zamiast zapisu V[i,j] - czyli zapisu położenia neuronu w sieci, będziemy zapisywać Vij jako opis odpowiadającego pola macierzy.

Sieć skonstruowano w ten sposób, że dwa różne neurony (i,j) i (k,l), (i,j,k,l∈{1,...,N}, są połączone ze sobą wtedy i tylko wtedy, gdy:

- neurony znajdują się w tym samym wierszu macierzy, tj. i=k, albo
- neurony znajdują się w tej samej kolumnie macierzy, tzn. j=l, albo
- neurony znajdują się na tej samej przekątnej macierzy tzn. i+j=k+l albo i-j=k-l.

W każdym z tych przypadków waga połączenia jest ujemna. We wszystkich pozostałych przypadkach, tzn. przy każdym innym wzajemnym położeniu neuronów (i,j) oraz (k,l) neurony te nie są połączone (czyli waga połączenia równa jest zero). Wszystkie połączenia są symetryczne. Dla tego przypadku funkcja energetyczna ma na przykład postać

$$E = \frac{A}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big[\Big(\sum_{k=1,k\neq j}^{N} v_{ik}\Big)v_{ij} + \Big(\sum_{k=1,k\neq i}^{N} v_{kj}\Big)v_{ij}\Big] + \frac{B}{2}\sum_{i=2}^{N}\sum_{j=1}^{i-1}\Big[\Big(\sum_{k=i-j+1,k\neq i}^{N} v_{k,k-i+j}\Big)v_{ij}\Big]$$
$$+ \frac{B}{2}\sum_{i=1}^{N}\sum_{j=i}^{N}\Big[\Big(\sum_{k=1,k\neq i}^{N+i-j} v_{k,k-i+j}\Big)v_{ij}\Big] + \frac{B}{2}\sum_{i=1}^{N}\sum_{j=N-i+1}^{N}\Big[\Big(\sum_{k=i+j-N,k\neq i}^{N} v_{k,i+j-k}\Big)v_{ij}\Big]$$
$$+ \frac{B}{2}\sum_{i=1}^{N-1}\sum_{j=i}^{N-i}\Big[\Big(\sum_{k=1,k\neq i}^{i+j-1} v_{k,i+j-k}\Big)v_{ij}\Big] + C\Big(\sum_{i=1}^{N}\sum_{j=1}^{N} v_{ij} - N_{-}\Big)^2$$

(6)

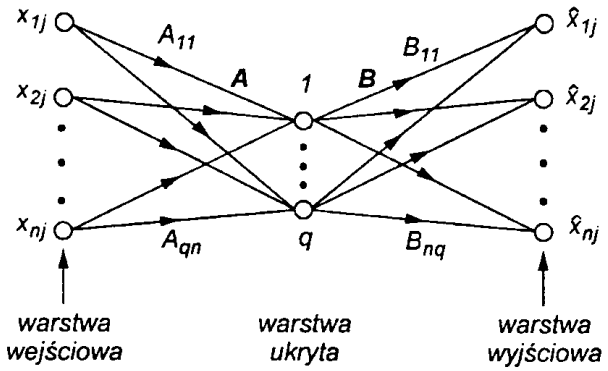
gdzie:
A,B i C są stałymi
N_ = N+𝜃 𝜃∈(0;1).

Przeprowadzono 150 testów symulacyjnych dla N=8, A=B=C=100 dla wartości N_ = 8.25, 8.50 i 8.75. Wybranie N=8 wynika w naturalny sposób z genezy problemu i jego praktycznej realizacji na standardowej szachownicy 64 polowej. Z dokładnej analizy problemu

wynika, że dobór parametru ma kluczowy wpływ na jakość wyników. Najlepsze okazały się dla wartości z przedziału <0;1).

3.1 Kompresja danych

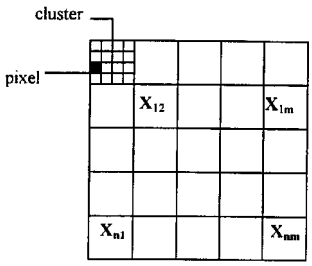
Zadanie kompresji danych polega na zmniejszeniu ilości informacji przechowywanej lub przesyłanej, przy zachowaniu możliwie jej pełnego odtworzenia (dekompresji). Stosuje się tutaj różne modele sieci i różne algorytmy ich uczenia a następnie kompresji. Jednym z typowych rozwiązań jest warstwowa struktura pokazana na rys. 6.



Rys. 3.2. Sieć warstwowa do kompresji danych

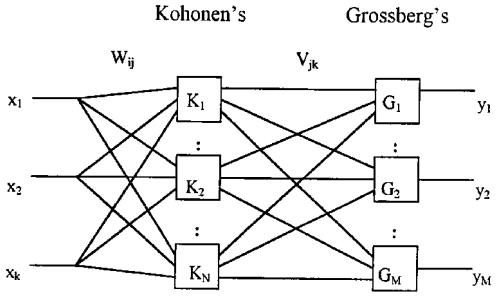
Dane wejściowe złożone z ciągu N wektorów n-elementowych (np. n=64) podawane są sekwencyjnie do warstwy wejściowej. Liczba elementów składowych wektora wejściowego jest równa liczbie 8x8 skwantyzowanych próbek oryginalnego obrazu. Jeżeli przyjmiemy, że każdy z elementów składowych wektora wejściowego ma poziom szarości w skali od 0 (całkowicie ciemny) do 255 (całkowicie jasny) to poziom szarości możemy zakodować przy pomocy jednego bajta (8-bitów) informacji. Warstwa ukryta zawiera q neuronów (q<<n, np. q=16) i realizuje kompresję danych. W omawianym przypadku następuje czterokrotna redukcja próbek oryginalnego obrazu. Warstwę wejściową i warstwę ukrytą sieci można tutaj przyrównać do nadajnika w układzie transmisji. W warstwie wyjściowej występuje ponownie n neuronów, i tutaj występuje dekompresja, czyli możliwie najwierniejsze odtworzenie oryginalnych 64 elementowych próbek obrazu. Warstwę wyjściową można tu porównać do odbiornika w systemie transmisji. Do nauczenia sieci typowo wykorzystuje się algorytm propagacji wstecznej.

Wstępne podzielenie obrazu (rys.7) na próbki czy „ramki” (clustery) opisane wektorami jest typowym rozwiązaniem przy kompresji danych.



Rysunek 3.3. Podział rysunki na ramki

Bardzo ciekawym i jednocześnie niesłychanie prostym w działaniu jest wykorzystanie do kompresji obrazów zmodyfikowanej sieci typu counter-propagation. Sieć tego typu składa się z dwóch warstw - warstwy Kohonena i warstwy Grossberga (rys.8).



Rysunek 3.4 Sieć counter-propagation

Pomiędzy warstwami sieci dany jest pełny zestaw połączeń. Warstwa Kohonena jest dołączona do wejścia sieci i realizuje zasadę WTA - winner takes all. Dla znormalizowanego sygnału wejściowego X każdy neuron Kohonena otrzymuje sygnał

$$NET_i = \sum_{j=1}^{k} w_{ij} x_j \qquad (7)$$

gdzie:
wij jest wagą połączenia pomiędzy j-tym a i-tym neuronem Kohonena.

Neuron o największym sygnale wejściowym generuje na swoim wyjściu sygnał równy 1 (blokując pozostałe neurony Kohonena) i pobudzając neurony Grossberga poprzez wagi vij.

Każdy neuron Grossberga otrzymuje na swoim wejściu sygnał

$$G_i = \sum_{j=1}^{N} v_{ij} K_j \qquad (8)$$

Ponieważ tylko jeden neuron Kohonena jest aktywny, warstwa Grossberga na swoim wyjściu generuje sygnał (wektorowy) określający wartości wag pomiędzy tym neuronem Kohonena i neuronami Grossberga. Każdy z elementów warstwy Grossberga realizuje funkcję outstar a liczba neuronów zależy od liczby neuronów Kohonena. Na wyjściu warstwy Grossberga otrzymujemy wektor Y skojarzony z odpowiednią klasą.

Przed kompresją obrazu najpierw należy nauczyć sieć. Obraz uczący dzielony jest na ramki złożone z pixeli o odcieniach szarości od 0 do 63. Cały obraz jest więc reprezentowany przez n*m wektorów k elementowych (k-liczba pixeli w ramce). W pierwszym etapie generujemy wagi w warstwie Kohonena. Początkowo mamy tylko jeden neuron Kohonena a jego wagi są identyczne ze składowymi pierwszego wektora uczącego. Dla następnych wektorów uczących obliczany jest iloczyn $\mathbf{X}^i\mathbf{W}_k$ (Xi jest i-tym wektorem uczącym, Wk są wagami k-tego neuronu). Jeżeli

$$(1 - X^i W_k) < B \qquad (9)$$

gdzie B (B∈(0;1)), określa różnicę pomiędzy wektorami uczącymi, wówczas i-ty wektor uczący zostanie zaliczony do danej klasy k, a wagi k-tego neuronu zmodyfikowane. Gdy warunek ten nie jest spełniony wprowadzony zostanie nowy neuron Kohonena o wagach równych składowym wektora uczącego. Po zakończeniu procedury dla wszystkich wektorów uczących są obliczane końcowe (uśrednione) wagi dla każdego neuronu. Teraz następuje uczenie neuronów w warstwie Grossberga - uczenie pod nadzorem. Dla każdego neuronu Kohonena generowane są wagi będące binarnym zapisem jego numeru. (Na przykład dla neuronu 6 mamy d6 = [0 1 1 0 0 0 0] a nowe wagi w warstwie Grossberga zgodnie z regułą

$$vik(t+1) = vik(t) + h[di - vik(t)] \qquad (10)$$

gdzie h jest współczynnikiem uczenia a vik wagą połączenia między k-tym neuronem Kohonena a i-tym neuronem Grossberga.

Obrazy podlegające kompresji są identycznie dzielone na ramki, tworzone są znormalizowane wektory reprezentujące te ramki. Te wektory są sygnałami wejściowymi sieci. Warstwa Kohonena określa „zwycięzcę”, który pobudza warstwę Grossberga generującą wektorowy sygnał będący zapisanym w kodzie ASCII numerem neuronu Kohonena. Ten wektor razem z oryginalną długością wektora wejściowego jest zapisywany. Dekompresja jest procedurą odwrotną. Ze zbioru odczytujemy numer wektora wraz z oryginalną długością aby odtworzyć ramkę.

Symulacja komputerowa pokazała nadspodziewanie dobre działanie modelu. Dla obrazu o wymiarach 80*80 pixeli podzielonego na ramki 2*2 (1600 wektorów) i B= 0.1 potrzebny był tylko 1 neuron Kohonena (!!!) przy współczynniku kompresji 2. Dla B=0.0016 i ramek 4*4 (400 wektorów) potrzebnych było 256 neuronów Kohonena a otrzymany współczynnik kompresji wynosił 8.

Łatwo można pokazać, że w zakresie do 256 neuronów Kohonena wzrost ich liczby prowadzi do poprawy obrazu (po dekompresji) przy stałym współczynniku kompresji równym 2 (dla ramek o wymiarze k=2*2) i współczynniku kompresji równym 8 (dla ramek o wymiarze k=4*4). Zwiększając liczbę neuronów Kohonena powyżej 256 zmniejsza współczynnik kompresji odpowiednio z k/2 do 3k/8. Zwiększanie k wymaga bardzo znacznego zwiększenia liczby neuronów aby zachować dobrą jakość kompresji.

4. Algorytmy uczenia sieci

4.1 Reguła uczenia DELTA

Niech czynnik ↱ oznacza różnicę pomiędzy wymaganą wielkością sygnału wyjściowego k-tego elementu warstwy wyjściowej d_k a aktualną (rzeczywistą) wielkością OUTk.

Przyjmijmy, że średni błąd kwadratowy E określony jest jako

$$E = \sum_{k=1}^{N} \delta_k^2 = \sum_{k=1}^{N} (d_k - OUT_k)^2 \qquad (11)$$

Ponieważ

$$OUT_k = \sum_{i=1}^{n} w_{ik} x_i$$

więc

$$E = \sum_{k=1}^{N} (d_k - \sum_{i=1}^{n} w_{ik} x_i)^2 \quad (12)$$

Funkcja błędu jest funkcją kwadratową ze względu na wagi (niewiadome). Jako funkcja kwadratowa ma dokładnie jedno minimum, a zatem funkcja błędu ma jedno minimum ze względu na każdą wagę.

Aby znaleźć to minimum posłużymy się metodą malejącego gradientu (gradient descend method).

Metoda ta mówi, że zmiana wartości zmiennej jest proporcjonalna do pochodnej funkcji błędu (ze względu na tą zmienną), wziętą ze znakiem minus

$$\Delta w_i = - \eta \frac{\partial E}{\partial w_i} \qquad (13)$$

gdzie η jest współczynnikiem proporcjonalności. W ten sposób waga może zostać zmieniona (dopasowana).

Obliczamy pochodną funkcji błędu E

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial E}{\partial \delta_k} \frac{\partial \delta_k}{\partial w_{ik}} = 2\delta_k \frac{\partial \delta_k}{\partial OUT_k} \frac{\partial OUT_k}{\partial w_{ik}} = 2\delta_k (-1) x_i = -2\delta_k x_i \qquad (14)$$

i stąd

$$\Delta w_{ik} = 2\eta \delta_k x_i \qquad (15)$$

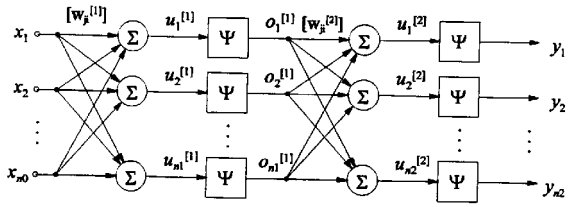
Reguła DELTA zmienia wagi w sieci proporcjonalnie do błędu na wyjściu sieci (różnicy pomiędzy rzeczywistym sygnałem wyjściowym a sygnałem wzorcowym) i wielkości odpowiedniej sygnału wejściowego.

4.2 Algorytm propagacji wstecznej

Algorytm propagacji wstecznej jest jednym z najskuteczniejszych algorytmów uczenia sieci wielowarstwowej. Jego nazwa pochodzi od sposobu obliczania błędu w poszczególnych warstwach sieci. Najpierw obliczane są błędy w warstwie ostatniej na podstawie porównania aktualnych i wzorcowych sygnałów wyjściowych i na tej podstawie dokonywane są zmiany wag połączeń, następnie w warstwie ją poprzedzającej i tak dalej aż do warstwy wejściowej. W algorytmie propagacji wstecznej można wyróżnić trzy fazy:

- podanie na wejście sygnału uczącego x i wyliczenie aktualnych wyjść y.
- porównujemy sygnał wyjściowy y z sygnałem wzorcowym d, a następnie wyliczamy lokalne błędy dla wszystkich warstw sieci,
- adaptacja wag.

Opisany proces kontynuuje się aż do momentu kiedy sygnały wyjściowe sieci będą dostatecznie bliskie oczekiwanym. Celem algorytmu propagacji wstecznej jest minimalizacja funkcji energetycznej. Dla jasności opisu zaprezentujemy opis algorytmu dla sieci składającej się z warstwy wejściowej, ukrytej i wyjściowej, co nie zmniejszy ogólności rozważań. Zakładamy, że warstwa wejściowa ma no elementów, warstwa ukryta n1 a warstwa wyjściowa n2 elementów (rys. 11)



Sieć neuronowa jednokierunkowa dwuwarstwowa.

Rysunek 4.1 Siec neuronowa jednokierunkowa dwuwarstwowa

Funkcja energetyczna dla danego wzorca uczącego jest zdefiniowana jako błąd średniokwadratowy.

$$E_p = \frac{1}{2}\sum_{j=1}^{n_2}(d_{jp} - y_{jp})^2 \qquad (16)$$

gdzie djp i yjp są odpowiednio wzorcowym i aktualnym sygnałem wyjściowym j-tego elementu warstwy wyjściowej przy wymuszeniu sygnałem uczącym p.

Metodę uaktualniania wag dla warstwy wyjściowej można zapisać następująco

$$\Delta w_{ji}^{[2]} = -\eta\frac{\partial E_p}{\partial w_{ji}^{[2]}} = -\eta\frac{\partial E_p}{\partial u_j^{[2]}}\frac{\partial u_j^{[2]}}{\partial w_{ji}^{[2]}} \qquad (17)$$

gdzie indeks (.)[2] oznacza warstwę wyjściową.

Biorąc pod uwagę, że

$$u_j^{[2]} = \sum_{i=1}^{n_2} w_{ji}^{[2]} o_i^{[1]} \qquad (18)$$

oraz definiując błąd lokalny jako

$$\delta_j^{[2]} = -\frac{\partial E_p}{\partial u_j^{[2]}} = -\frac{\partial E_p}{\partial e_{jp}}\frac{\partial e_{jp}}{\partial u_j^{[2]}} = -e_{jp}\frac{\partial (d_{jp} - y_{jp})}{\partial u_j^{[2]}} = e_{jp}\frac{\partial y_{jp}}{\partial u_j^{[2]}} = e_{jp}\frac{\partial \psi}{\partial u_j^{[2]}} \qquad (19)$$

otrzymujemy zależność opisującą aktualizację wag w warstwie wyjściowej

$$w_{ji}^{[2]}(t) = w_{ji}^{[2]}(t-1) + \Delta w_{ji}^{[2]} \qquad (20)$$

$$\Delta w_{ji}^{[2]} = \eta \delta_j^{[2]} o_i^{[1]}$$

Znalezienie błędu lokalnego dla elementów w warstwach ukrytych jest problemem znacznie trudniejszym, gdyż nie znany jest poprawny sygnał wyjściowy z elementów tej warstwy. Dlatego korzysta się z tych danych, które są dostępne lub łatwo mogą być policzone

$$\delta_j^{[1]} = -\frac{\partial E_p}{\partial u_j^{[1]}} = -\frac{\partial E_p}{\partial o_{jp}^{[1]}} \frac{\partial o_{jp}^{[1]}}{\partial u_j^{[1]}} = -\frac{\partial E_p}{\partial o_{jp}^{[1]}} \frac{\partial \psi}{\partial u_j^{[1]}} \qquad (21)$$

gdzie

$$\frac{\partial E_p}{\partial o_j^{[1]}} = \sum_{i=1}^{n2} \frac{\partial E_p}{\partial u_i^{[2]}} \frac{\partial u_i^{[2]}}{\partial o_j^{[1]}} = \sum_{i=1}^{n2} \left(\frac{\partial E_p}{\partial u_i^{[2]}} \right) \frac{\partial}{\partial o_j^{[1]}} \left(\sum_{i=1}^{n2} w_{ik}^{[2]} o_k^{[1]} \right) = -\sum_{i=1}^{n2} \delta_i^{[2]} w_{ij}^{[2]} \qquad (22)$$

a w konsekwencji wzór na błąd lokalny otrzymujemy w postaci

$$\delta_j^{[1]} = \frac{\partial \psi}{\partial u_j^{[1]}} \sum_{i=1}^{n2} \delta_i^{[2]} w_{ij}^{[2]} \qquad (23)$$

i na poprawkę wagi

$$\Delta w_{ji}^{[1]} = \eta \delta_j^{[1]} x_i$$

W przypadku większej liczby warstw ukrytych ostatnie wzory nie ulegają zmianie, a jedynie odpowiednie indeksy.

5. Sieci neuronowe w zastosowaniu do algebry macierzowej

Sieci neuronowe jednokierunkowe (feedforward) mogą być łatwo wykorzystane w algebrze macierzowej realizując standardowe operacje macierzowe jak na przykład: odwracanie macierzy, mnożenie macierzy, obliczanie wartości i wektorów własnych macierzy, dekompozycja macierzy itp. Algorytmy te wykorzystując równoległość przetwarzania pozwalają na uzyskanie dużej szybkości działania rzędu mikrosekund, czyli praktycznie działania w czasie rzeczywistym. Jak i poprzednio podstawowym problemem jest właściwe określenie funkcji energetycznej, której minimalizacja pozwala stworzyć odpowiednią strukturę sieci.

5.1 Wyznaczenie macierzy odwrotnej

Załóżmy, że dana jest nieosobliwa macierz kwadratowa A rzędu n. Chcemy zaprojektować sieć neuronową obliczającą macierz B = A-1. Macierz odwrotna B spełnia z definicji następujące równanie macierzowe

$$BA = 1 \qquad (24)$$

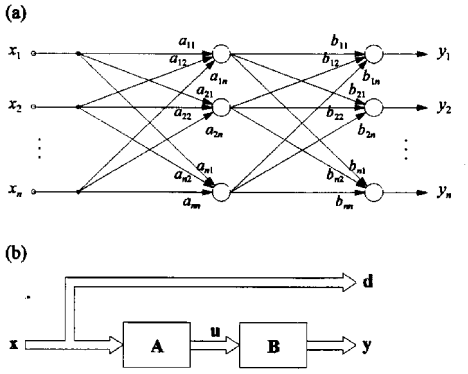
Mnożąc obustronnie przez dowolny niezerowy wektor x = [x1(t),x2(t),...xn(t)] otrzymujemy po przekształceniu

$$BAx - x = 0 \qquad (25)$$

Wprowadzając definicję funkcji energetycznej w postaci

$$\mathbf{E} = \frac{1}{2}(\|\mathbf{BAx} - \mathbf{x}\|_2)^2 \qquad (26)$$

proces rozwiązania równania macierzowego (25) można zastąpić minimalizacją funkcji (26). Proces uczenia tej sieci łatwo można zinterpretować przez pokazanie grafów przepływu sygnałów. Na rysunku 12a przedstawiono graf a na rys.12b postać blokową przepływu sygnałów odpowiadającą zdefiniowanej powyżej funkcji energetycznej (26). Wartości wag połączeń Aij macierzy A są wagami stałymi nie podlegającymi uczeniu natomiast wagi Bij macierzy poszukiwanej macierzy odwrotnej B podlegają uczeniu zgodnie z algorytmem propagacji wstecznej.



Rysunek 5.1 Sieć do odwracania macierzy a) graf, b) postać blokowa

Ponieważ po zakończeniu procesu uczenia spełnione są następujące zależności

$$y = Bu = BAx = x \qquad (27)$$

to dowolny niezerowy wektor x pełni tu podwójną rolę: jest wektorem uczącym, będącym sygnałem wejściowym sieci a jednocześnie wektorem, jaki ma być na wyjściu sieci. (Jest to więc sieć autoasocjacyjna).

Pomimo, że schemat sieci pokazany na rys.12 zawiera dwie warstwy, z punktu widzenia algorytmu uczącego jest to sieć jednowarstwowa. Jako algorytmu uczącego można użyć metodę największego spadku, i wówczas

$$b_{ij}(t) = b_{ij}(t - 1) + \eta(x_{ip} - y_{ip})u_{jp} \qquad (28)$$

gdzie:
p jest numerem wzorca uczącego.

5.2 Mnożenie macierzy

Jeżeli macierz C jest równa iloczynowi macierzy A i B to spełnione jest następujące równanie macierzowe

$$C = AB \qquad (29)$$

Aby skonstruować sieć neuronową zdolną do rozwiązania tego problemu musimy określić funkcję energetyczną, odpowiadającą powyższemu równaniu i przyjmującą minimum kiedy to równanie jest spełnione.

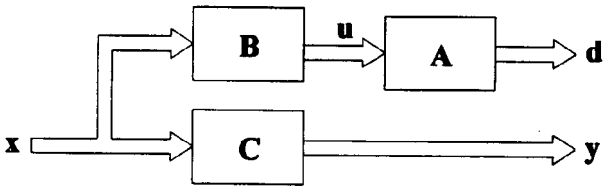
Zgodnie z techniką przyjętą w poprzednim rozdziale, obie strony równania mnożymy przez dowolny niezerowy wektor x. Po przekształceniu otrzymujemy

$$ABx - Cx = 0 \qquad (30)$$

Na podstawie tej zależności definiujemy funkcję energetyczną dla zadania mnożenia macierzy:

$$\mathbf{E} = \frac{1}{2}\left(\|\mathbf{ABx} - \mathbf{Cx}\|_2\right)^2 \qquad (31)$$

Sieć skonstruowana na tej podstawie jest siecią jednowarstwową, mino że na schemacie blokowym (rys.13) umieszczone są trzy warstwy sieci neuronowej.



Rysunek 5.2 Schemat blokowy sieci do mnożenia macierzy

Spośród trzech warstw uczeniu podlega jedynie warstwa odpowiadająca macierzy C realizująca zależność

$$y = Cx \qquad (32)$$

Ponieważ po zakończeniu procesu uczenia sieć powinna spełniać równanie (29) w sieci są dwie dodatkowe warstwy o stałych wagach (odpowiednio elementy macierzy A i B), które służą do wyliczenia wektora wzorcowego d, zgodnie z zależnością

$$d = Au = ABx \qquad (33)$$

Jako algorytm uczący tutaj także można wykorzystać metodę największego spadku. Zgodnie z nią reguła adaptacyjna ma postać

$$c_{ij}(t+1) = c_{ij}(t) + \eta(d_{ij} - y_{ip})x_{jp} \qquad (34)$$

gdzie:

p jest numerem wzorca uczącego.

5.3. Rozkład LU macierzy

Ponieważ rozkład LU macierzy kwadratowej A na macierze trójkątną dolna L i trójkątną górną U takie że:

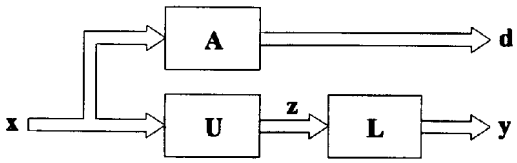
$$A = LU \qquad (35)$$

w ogólnym przypadku nie jest jednoznaczny, ograniczymy naszą dyskusję do przypadku kiedy macierz trójkątna dolna L ma na przekątnej wartości 1, co pozwala na uzyskanie jednoznaczności rozkładu.

Po pomnożeniu zależności (35) przez dowolne niezerowy wektor x i po przekształceniach, funkcję energetyczną dla rozkładu LU definiujemy jako

$$E = \frac{1}{2}(\|LUx - Ax\|_2)^2 \qquad (36)$$

Dwuwarstwowa się zaprojektowana na podstawie tej funkcji energetycznej jest bardziej skomplikowana niż w przypadku odwracania czy mnożenia macierzy. Tutaj obydwie warstwy (rys.14), odpowiadające macierzom L i U podlegają uczeniu.



Rysunek 5.3 Schemat blokowy sieci do rozkładu LU macierzy

Sieć realizuje zależność

y = Lz = LUx (37)

a zgodnie z rozkładem (35), wzorcowa odpowiedź jest równa

d = Ax (38)

w sieci znajduje się równoległa warstwa pomocnicza o ustalonych wagach, liczbowo odpowiadających elementom macierzy A służąca do wyliczania wektora d.

Macierze L i U muszą spełniać przyjęte warunki, więc wago lii mają ustaloną wartość 1 – natomiast wagi odpowiadające odpowiednim elementom macierzy trójkątnych L i U są równe zero.

Minimalizację funkcji (36) przeprowadzamy metoda propagacji wstecznej, i mamy:

dla i>j, oraz

$$l_{ij}(t+1) = l_{ij}(t) + \eta e_{ip} z_{jp} \qquad (39)$$

$$u_{ij}(t+1) = u_{ij}(t) + \eta\left[\sum_{h=1}^{n} l_{hi}(t+1)e_{hp}\right] x_{jp} \qquad (40)$$

dla i ≠ j

gdzie:

$$e_{ip} = d_{ip} - y_{ip} \qquad d_{ip} = \sum_{j=1}^{n} a_{ij} x_{jp}$$

$$y_{ip} = \sum_{j=1}^{in} l_{ij} z_{jp}$$

a p jest numerem wzorca uczącego.

5.4. Poszukiwanie wartości i wektorów własnych macierzy

Przyjmijmy, że macierz kwadratowa A rzędu n jest macierzą symetryczną. Wartości własne tej macierzy definiowane są jako zera wielomianu

$$\det(\lambda - A) = 0 \qquad (41)$$

tworzą macierz diagonalną wartości własnych L = diag[λ_1, λ_2,..., λ_n], w której wszystkie wartości własne λ są rzeczywiste. Wektor własny Vi odpowiadający wartości własnej λ jest n-wymiarowym wektorem kolumnowym Vi = [V1i, V2i,..., Vni]T, który jest rozwiązaniem układu równań algebraicznych

$$(A - \lambda_1)Vi = 0 \qquad (42)$$

(Dla jednoznaczności rozwiązania można przyjąć, że wektory własne mają znormalizowaną długość). Przy założeniu jednokrotnych wartości własnych macierz A można zdekomponować do postaci

$$A = VLV\text{-}1 \qquad (43)$$

w której macierz V składa się z poszczególnych wektorów własnych uporządkowanych kolumnowo. Dla symetrycznej macierzy A, V jest macierzą ortogonalną (V-1 = VT).

Rozwiązanie problemu wartości własnych polega na określeniu macierzy L i V w taki sposób, aby spełnione były równania

$$VLVT - A = 0 \qquad (44)$$

oraz

$$VTV - 1 = 0 \qquad (45)$$

Przekształcamy te równania do postaci

$$VLVTX - AX = 0 \qquad (46)$$

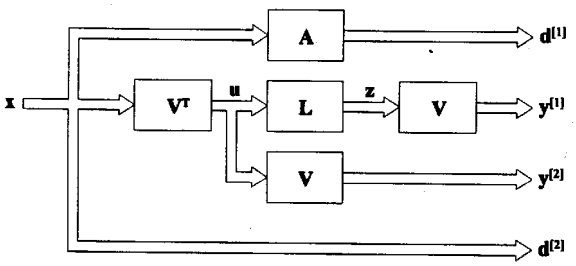
oraz

$$VTVX - X = 0 \qquad (47)$$

i rozwiązania problemu będziemy poszukiwać przez minimalizację funkcji energetycznej zdefiniowanej jako

$$E = \frac{1}{2}\left((\left\| VLV^T X - AX \right\|_2)^2 + (\left\| V^T VX - X \right\|_2)^2 \right) \qquad (48)$$

W tym wyrażeniu X, (dowolny niezerowy wektor), jest wejściowym sygnałem uczącym. Zależności macierzowe VLVTX oraz VTVX reprezentują sobą dwa wektory wyjściowe sieci a zależności AX oraz X - odpowiadające im wektory zadane. Sieć neuronową odpowiadającą powyższej funkcji energetycznej można traktować jako złożenie dwu sieci odpowiadających odpowiednim członom funkcji energetycznej: heteroasocjacyjnej o parze uczącej (X, AX) oraz autoasocjacyjnej o wektorze wejściowym (i wyjściowym) X. Aktualna wartość wyjścia sieci heteroasocjacyjnej Y(1) = VTLVX a części autoasocjacyjnej Y(2) = VTVX. Przepływ sygnałów w sieci pokazany jest na rys. 15.



Rysunek 5.4 Schemat blokowy sieci do poszukiwania wartości i wektorów własnych macierzy

Przy zadanym wektorze uczącym x, dla podsieci odpowiedzialnej za diagonalizację macierzy A, odpowiada wzorcowa odpowiedź D[1]=Ax, natomiast dla drugiej podsieci odpowiedź d[2] = x. Pomimo, ze w sieci występują cztery warstwy ze zmiennymi wagami reguły uczenia trzeba określić tylko dla dwóch warstw odpowiadającym macierzom L i V, ponieważ macierz V reprezentowana jest przez trzy warstwy sieci

bd[1] = Ax, y[1] = Vz = VLu = VLVTx

d[2] = x, y[2] = Vu = VVTx (49)

Odpowiednie wzory uczące (zmiany wag) można przedstawić w postaci

$$v_{ij}(t+1) = v_{ij}(t) + \eta\left(e_{ip}^{[1]} z_{jp} + e_{ip}^{[2]} u_{jp}\right) \qquad (50)$$

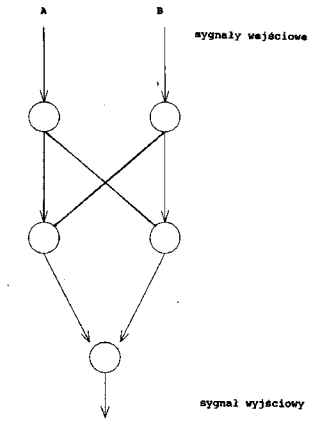
$$\lambda_i(t+1) = \lambda_i(t) + \eta\left(\sum_{k=1}^{n} e_{kp}^{[1]} v_{ki}\right) u_{ip} \qquad (51)$$

gdzie

$$e_{ip}^{[1/2]} = d_{ip}^{[1/2]} - y_{ip}^{[1/2]}$$

5.5. Sieci neuronowe realizujące operacje logiczne

Większość prac na temat sieci neuronowych zajmuje się takimi zagadnieniami, jak rozpoznawanie obrazów czy pamięci skojarzeniowe. Jednym z ciekawych, a niezbyt jeszcze dokładnie przebadanych zastosowań, jest użycie sieci neuronowych do realizacji funkcji logicznych wielu zmiennych. Dawno temu, M. Minsky i S. Pappert przy omawianiu perceptronu, a właściwie przy wykazywaniu jego wad, użyli funkcji XOR jako przykładu operacji nie dającej się zrealizować w jednowarstwowym perceptronie. Tą prostą funkcją logiczną można zrealizować za pomocą sieci neuronowych na wiele różnych sposobów. Jedna z najprostszych struktur pokazana jest na rys. 16. Połączenia oznaczone strzałką mają dodatnią wagę równą +1, natomiast połączenia bez strzałek mają wagę równą -1. Wszystkie elementy mają taką samą charakterystykę nieliniową z progiem równym zero. Sygnały wejściowe mogą przyjmować wartości równe zero albo jeden.



Rysunek 5.5 Graf sieci do realizacji operacji logicznej XOR

Jest to przykład sieci zdolnej do realizacji jednej, konkretnej funkcji logicznej, ale z tego typu elementów można zbudować sieci, które będą zdolne do realizacji bardziej złożonych operacji.

Każdą funkcję logiczną dwóch zmiennych można opisać zależnością

$$f(A,B) = \alpha_0 \bar{A}\,\bar{B} + \alpha_1 A\,\bar{B} + \alpha_2 \bar{A}\,B + \alpha_3 A\,B \tag{52}$$

zwaną postacią kanoniczną zmiennych logicznych A i B.

Przyjmując odpowiednie wartości współczynników α_i, i = 0,1,2,3, równe zero albo jeden - otrzymujemy dowolną z 16 funkcji tych zmiennych. Przykładowo przyjmując

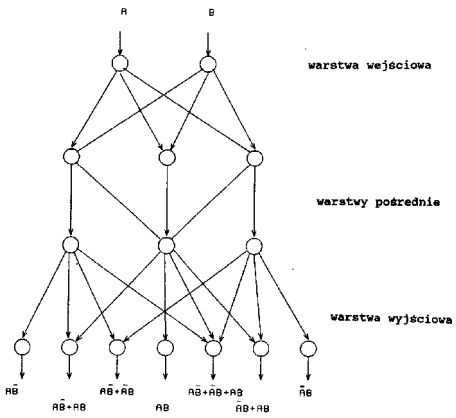
$$\alpha_1 = \alpha_2 = 1 \text{ i } \alpha_0 = \alpha_3 = 0$$

otrzymujemy

$$f(A,B) = A\,\bar{B} + \bar{A}\,B \tag{53}$$

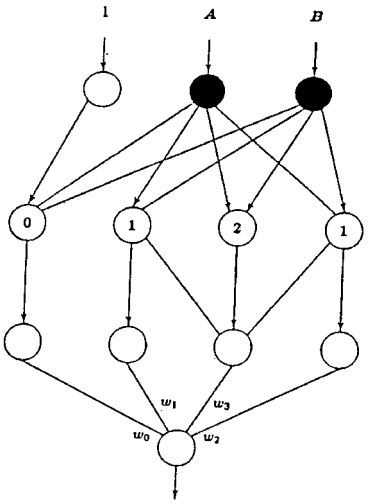
czyli funkcję XOR. Zapis ten można uogólnić na zagadnienie dowolnej funkcji logicznej n-zmiennych.

Na rysunku 17 pokazano sieć o binarnych (0 albo 1) sygnałach wejściowych. Każdy z elementów wyjściowych sieci realizuje inną funkcję logiczną dwóch zmiennych. Jak wiadomo, dla dwóch zmiennych mamy ogółem 16 różnych funkcji logicznych. Sieć na rys. 17 pokazuje 7 spośród tych 16, a mianowicie te, w których różny od zera sygnał wejściowy generuje różny od zera sygnał wyjściowy.



Rysunek 5.6 Sieć realizująca operacje logiczne

Na rysunku 18 pokazana jest uniwersalna sieć zdolna do zrealizowania każdej z 16 dwu-elementowych funkcji logicznych. Aby to uzyskać, należy przyjąć wartości wag w_i, i = 0,1,2,3, takie same jak odpowiednie wartości współczynników α_i, i = 0,1,2,3, we wzorze (52). Model ten można zresztą również uogólnić na przypadek n-zmiennych, przy czym, stopień komplikacji sieci wzrasta w dosyć znaczny sposób.



Rysunek 5.7 Sieć realizacji operacji logicznych

6. Podsumowanie

Przedstawiłem zaledwie kilka spośród wielu obszarów techniki, w których stosuje się lub można zastosować sieci neuronowe.

W ostatnim 10-leciu ponownie wzrosło zainteresowanie sieciami neuronowymi, sieci stały się modne. W bardzo wielu ośrodkach naukowych trwają badania nad ich właściwościami i możliwościami ich wykorzystania. Corocznie drukuje się setki prac naukowych i dziesiątki książek poświęconych tej tematyce. Istnieją specjalne czasopisma poświęcone wyłącznie sieciom, corocznie odbywają się dziesiątki konferencji i seminariów.

Pojawia się pytanie co dalej? Jakie są perspektywy na najbliższe i dalsze lata? Jakiej techniki, jakich metod realizacji powinniśmy użyć aby stworzone modele teoretyczne spełniły pokładane w nich nadzieje?

Na te i wiele innych pytań odpowiedzi powinny udzielić najbliższe lata. Wierzymy, że sztuczne sieci neuronowe, ta „nędzna” namiastka systemu nerwowego jak i neurokomputery - równie „nędzna” imitacja mózgu pozwolą kiedyś na uzyskanie nowych i ciekawych wyników.

Literatura

[1] R.Beale, T.Jackson, Neural Computing, An Introduction, A.Hilger IOP Publ. Co. Bristol 1990.

[2] T.Berus, B.Macukow, Od sieci neuronowych do neurokomputerów I, Informatyka, 4, 14-17, 1991.

[3] T.Berus, B.Macukow, Od sieci neuronowych do neurokomputerów II, Informatyka, 1, 1-7, 1993.

[4] R.D.Brandt, Y.Wang, A.J.Laub, S.K.Mitra, Alternative Networks for Solving the Traveling Salesman Problem and List-Matching Problem, INCC II, 333-339, 1988.

[5] A.Cichocki, R.Unbehauen, Neural Networks for Optimization and Signal Processing, J.Wiley and Sons, 1993.

[6] J.Hertz, A.Krogh, R.G.Palmer, Wstęp do teorii obliczeń neuronowych, WNT, Warszawa 1993.

[7] J.J.Hopfield, Neural Networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad,. Sci. USA, 79, 2554-2558, 1982.

[8] J.J.Hopfield, D.W.Tank, „Neural" Computation and Decisions in Optimization Problems, Biol. Cyber. 52, 141-152, 1985.

[9] J.Korbicz, A.Obuchowicz, D.Uciński, Sztuczne sieci neuronowe, podstawy zastosowania, Akad. Ofic, Wyd. PLJ, Warszawa 1994.

[10] Lippmann, An Introduction to Computing with Neural Nets, IEEE ASSP Magasin, 4-22, April 1987.

[11] B.Macukow, Neurokomputery, Postępy Fizyki, 41, 265-284, 1990.

[12] B.Macukow, H.H.Arsenault, Logic Operations with Neural Networks, Control and Cybernetics, 20, 115-133, 1991.

[13] J.Mańdziuk, B.Macukow, A Neural Network Designed to Solve the N-Queens Problem, Biol. Cyber. 66, 375-379, 1992.

[14] S.Osowski, Sieci neuronowe, Ofic. Wyd. Pol. Warszawskiej, Warszawa 1994.

[15] R.Tadeusiewicz, Sieci neuronowe, Akad. Oficyna Wyd. RM., Warszawa 1993.

[16] J. Żurada, M.Barski, W.Jędruch, Sztuczne sieci neuronowe, Wyd.Nauk. PWN, Warszawa, 1996.

Zawartość dysku

Informacja o zawartości dysku powinna być podana w sposób ogólny, przykładowo:

1. Tekst pracy w formacie .pdf
2. Pliki źródłowe aplikacji "Dekoder kodów QR"
3. Instrukcja instalacji i obsługi programu

Załącznik 1

Załączniki powinny zawierać informacje o charakterze zbyt szczegółowym z punktu widzenia opisu w tekście. Na przykład, mogą to być karty katalogowe układów użytych do realizacji pracy lub diagramy UML opisujące strukturę aplikacji.

Państwowa Uczelnia im. Stefana Batorego

Imię i nazwisko: .....
Numer albumu: .....
Kierunek: .....

OŚWIADCZENIE

Świadomy/a odpowiedzialności oświadczam, że złożona przeze mnie praca dyplomowa/projekt dyplomowy licencjacki/inżynierski* pt.

.....

.....

.....

został/a napisana samodzielnie w oparciu o zgromadzoną literaturę ujętą w bibliografii. Jednocześnie oświadczam, że w/w praca nie narusza praw autorskich w rozumieniu Ustawy z dnia 4.02.1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. nr 24 poz. 83 z późn zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony.

Wyżej wymieniona praca nie była także wcześniej podstawą żadnej urzędowej procedury nadania dyplomu wyższej uczelni lub tytułu zawodowego.

Skierniewice, dnia .....
.....

własnoręczny podpis

OŚWIADCZENIE

Wyrażam/nie wyrażam* zgodę na udostępnienie mojej pracy dyplomowej/projektu dyplomowego pt.

.....

.....

.....

Skierniewice, dnia .....
.....

własnoręczny podpis

* niepotrzebne skreślić