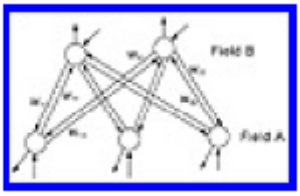
BAM: dwukierunkowa pamięć asocjacyjna

Wstęp

Model dwukierunkowej pamięci asocjacyjnej (BAM) ma sieć neuronową składającą się z dwóch warstw i jest w pełni połączony z każdą warstwą z drugą. Oznacza to, że istnieją połączenia zwrotne z warstwy wyjściowej do warstwy wejściowej. Jednak wagi połączeń między dowolnymi dwoma danymi neuronami z różnych warstw są takie same. Możesz nawet uznać, że jest to pojedyncze połączenie dwukierunkowe o jednej wadze. Macierz wag dla połączeń z warstwy wyjściowej do warstwy wejściowej jest po prostu transpozycją macierzy wag dla połączeń między warstwą wejściową i wyjściową. Jeśli oznaczymy macierz wag połączeń do przodu przez W, to WT jest macierzą wag dla połączeń warstwy wyjściowej z warstwą wejściową. Jak pamiętasz, transpozycję macierzy uzyskuje się po prostu przez zamianę wierszy i kolumn macierzy. Istnieją dwie warstwy neuronów, warstwa wejściowa i warstwa wyjściowa. Nie ma połączeń bocznych, to znaczy nie ma połączonych dwóch neuronów w tej samej warstwie. Połączenia powtarzające się, które są połączeniami zwrotnymi do neuronu od samego siebie, mogą być obecne lub nie. Architektura jest dość prosta. Rysunek 8.1 pokazuje układ tego modelu sieci neuronowej, wykorzystujący tylko trzy neurony wejściowe i dwa neurony wyjściowe.



Istnieją połączenia zwrotne z pola A do pola B i odwrotnie. Ten rysunek wskazuje również na obecność wejść i wyjść w każdym z dwóch pól dla dwukierunkowej sieci pamięci asocjacyjnej. Ciężary połączeń są również pokazane jako etykiety tylko na kilku połączeniach na tym rysunku, aby uniknąć bałaganu. Ogólny przypadek jest analogiczny.

Wejścia i wyjścia

Dane wejściowe do sieci BAM to wektor liczb rzeczywistych, zwykle w zbiorze { -1, +1 }. Wyjściem jest również wektor liczb rzeczywistych, zwykle w zbiorze { -1, +1 }, o takim samym lub innym wymiarze niż wejście. Te wektory można uznać za wzorce, a sieć tworzy heteroskojarzenia wzorców. Jeśli wyjście ma być takie samo jak wejście, to prosisz sieć o wykonanie autoasocjacji, co robi, i staje się to szczególnym przypadkiem ogólnej aktywności tego typu sieci neuronowej. W przypadku wejść i wyjść, które nie są poza zestawem zawierającym tylko -1 i +1, spróbuj wykonać następną procedurę. Możesz najpierw wykonać mapowanie na liczby binarne, a następnie mapowanie każdej cyfry binarnej na cyfrę dwubiegunową. Na przykład, jeśli twoje dane wejściowe są imionami osób, każdy znak w nazwie można zastąpić jego kodem ASCII, który z kolei można zmienić na liczbę binarną, a następnie każdą cyfrę binarną 0 można zastąpić przez -1. Na przykład kod ASCII dla litery R to 82, czyli 101010, jako liczba binarna. Jest to mapowane na dwubiegunową strunę 1 -1 1 -1 -1 1 –1. Jeśli nazwa składa się z trzech znaków, ich kody ASCII w postaci binarnej można połączyć lub zestawić i otrzymać odpowiedni ciąg dwubiegunowy. Ten dwubiegunowy ciąg można również traktować jako wektor znaków dwubiegunowych.

Wagi i trening

BAM nie modyfikuje ciężarów podczas swojego działania i jak wspomniano w rozdziale 6, podobnie jak sieć Hopfield, wykorzystuje trening jednorazowy. Adaptacyjna odmiana BAM, zwana adaptacyjną dwukierunkową pamięcią asocjacyjną (ABAM), przechodzi nadzorowany trening iteracyjny. BAM potrzebuje przykładowych par wektorów. Pary użyte jako wzorce to takie, które wymagają heteroasocjacji. Macierz wag, są dwie, ale jedna jest tylko transpozycją drugiej, jak już wspomniano, jest konstruowana w kategoriach przykładowych par wektorów. Użycie przykładowych wektorów to jednorazowa nauka — aby określić, jakie powinny być wagi. Po takim określeniu wag i przedstawieniu wektora wejściowego, potencjalnie skojarzony wektor jest wyprowadzany. Jest przyjmowany jako wejście w przeciwnym kierunku, a jego potencjalnie skojarzony wektor jest uzyskiwany z powrotem w warstwie wejściowej. Jeśli ostatni znaleziony wektor jest taki sam, jak pierwotnie wprowadzony, wówczas występuje rezonans. Załóżmy, że wektor B jest uzyskiwany na jednym końcu, w wyniku wprowadzenia C na drugim końcu. Jeśli z kolei B jest wprowadzane podczas następnego cyklu operacji na końcu, w którym zostało uzyskane, i wytwarza C na przeciwległym końcu, to masz parę heteroskojarzonych wektorów. To właśnie dzieje się w sieci neuronowej BAM. Poniżej przedstawiono równania do wyznaczenia macierzy wag, gdy k par przykładowych wektorów oznaczono przez (Xi, Yi), i w zakresie od 1 do k. Zauważ, że T w indeksie górnym macierzy oznacza transpozycję macierzy. Kiedy wymieniasz wiersze i kolumny, aby uzyskać transpozycję macierzy, piszesz wektor kolumn jako wektor wierszy i odwrotnie, aby uzyskać transpozycję wektora. Poniższe równania odnoszą się do par wektorów po zamianie ich składowych na wartości dwubiegunowe, tylko w celu uzyskania macierzy wag W. Po uzyskaniu W następuje dalsze wykorzystanie tych przykładowych wektorów w ich oryginalnej postaci.

$$\mathbf{W} = X_1^T Y_1 + \ldots + X_k^T Y_k$$

i

$$\mathbf{W}^T = Y_1^T X_1 + \ldots + Y_k^T X_k$$

Przykład

Załóżmy, że wybierasz dwie pary wektorów jako możliwe przykłady. Pozwól im być:

$X_1 = (1, 0, 0, 1)$, Y1= (0, 1, 1)

oraz

$X_2 = (0, 1, 1, 0)$, Y2 = (1, 0, 1)

Zamieniasz je w składowe dwubiegunowe i otrzymujesz odpowiednio (1, –1, –1, 1), (-1, 1, 1), (-1, 1, 1, –1) i (1, –1 , 1).

$$W = \begin{matrix} 1 \\ -1 \\ -1 \\ 1 \end{matrix} \begin{bmatrix} -1 & 1 & 1 \end{bmatrix} + \begin{matrix} -1 \\ 1 \\ 1 \\ -1 \end{matrix} \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} = \begin{matrix} -1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \\ -1 & 1 & 1 \end{matrix} + \begin{matrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{matrix} = \begin{matrix} -2 & 2 & 0 \\ 2 & -2 & 0 \\ 2 & -2 & 0 \\ -2 & 2 & 0 \end{matrix}$$

i

$$W^T = \begin{matrix} -2 & 2 & 2 & -2 \\ 2 & -2 & -2 & 2 \\ 0 & 0 & 0 & 0 \end{matrix}$$

Możesz pomyśleć, że ostatnia kolumna, w której W jest zerami, stanowi problem, ponieważ gdy na wejściu jest X_1 i niezależnie od wyniku, kiedy to wyjście jest prezentowane wstecz, nie daje X_1, który nie ma zera w ostatnim komponencie. Potrzebna jest tutaj funkcja progowania. Funkcja progowania jest przezroczysta, gdy wszystkie aktywacje mają wartość +1 lub –1. To wtedy po prostu wygląda na to, że robisz odwrotne mapowanie z wartości dwubiegunowych na binarne, co odbywa się przez zastąpienie każdego -1 przez 0. Gdy aktywacja wynosi zero, po prostu zostawiasz wyjście tego neuronu tak, jak było w poprzednim cyklu lub iteracji . Przedstawiamy teraz funkcję progową dla wyjść BAM

$$b_j|_{t+1} = \begin{cases} 1 & \text{if } y_j > 0 \\ b_j|_t & \text{if } y_j = 0 \\ 0 & \text{if } y_j < 0 \end{cases} \quad \text{and} \quad a_i|_{t+1} = \begin{cases} 1 & \text{if } x_i > 0 \\ a_i|_t & \text{if } x_i = 0 \\ 0 & \text{if } x_i < 0 \end{cases}$$

gdzie x_i i y_j są aktywacjami neuronów i i j odpowiednio w warstwie wejściowej i warstwie wyjściowej, a $b_j|t$ odnosi się do wyjścia neuronu j-tego w warstwie wyjściowej w cyklu t, natomiast $a_i|t$ odnosi się do wyjście neuronu i-tego w warstwie wejściowej w cyklu t. Zwróć uwagę, że na początku wartości a_i i b_j są takie same, jak odpowiadające im składniki w używanej przykładowej parze. Jeżeli neuronom wejściowym przedstawia się X_1 = (1, 0, 0, 1), to ich aktywacje podaje wektor (–4, 4, 0). Wektor wyjściowy, po zastosowaniu właśnie opisanej funkcji progowej, to (0, 1, 1). Ostatni składnik powinien być taki sam, jak wyjście w poprzednim cyklu, ponieważ odpowiadająca mu wartość aktywacji wynosi 0. Ponieważ X_1 i Y_1 to jedna przykładowa para, trzeci składnik Y_1 jest tym, czego potrzebujemy jako trzeci składnik wyjście w bieżącym cyklu pracy; dlatego nakarmiliśmy X_1 i otrzymaliśmy Y_1. Jeśli zasilimy Y_1 na drugim końcu (pole B), aktywacje w polu A wyniosą (2, –2, –2, 2), a wektorem wyjściowym będzie (1, 0, 0, 1), czyli X_1. Z wejściem pola A X_2 otrzymujesz aktywacje pola B (4, -4, 0), dając wektor wyjściowy jako (1, 0, 1), czyli Y_2. Zatem X_2 i Y_2 są ze sobą heteroskojarzone. Zmieńmy nasze X_1 na (1, 0, 1, 1). Wtedy macierz wag W wynosi

1 [-1 1 1] -1 [1 -1 1] -2 2 0

$$W = \begin{matrix} -1 \\ 1 \\ 1 \end{matrix} + \begin{matrix} 1 \\ -1 \end{matrix} = \begin{matrix} 2 & -2 & 0 \\ 0 & 0 & 2 \\ -2 & 2 & 0 \end{matrix}$$

i

$$W_T = \begin{matrix} -2 & 2 & 0 & -2 \\ 2 & -2 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{matrix}$$

To jest inny zestaw dwóch przykładowych par wektorów. Pary to X_1 = (1, 0, 1, 1), 1 = (0, 1, 1) i X_2 = (0, 1, 1, 0), Y_2 = (1, 0, 1). Oczywiście macierz wag jest inna, podobnie jak jej transpozycja. Jak wspomniano wcześniej, wagi nie zmieniają się podczas działania sieci, niezależnie od przedstawionych jej danych wejściowych. Wyniki są takie, jak pokazano w tabeli

Input vector	activation	output vector
$X_1 = (1, 0, 1, 1)$	(-4, 4, 2)	$(0, 1, 1) = Y_1$
$X_2 = (0, 1, 1, 0)$	(2, -2, 2)	$(1, 0, 1) = Y_2$
$Y_1 = (0, 1, 1)$	(2, -2, 2, 2)	$(1, 0, 1, 1) = X_1$
$Y_2 = (1, 0, 1)$	(-2, 2, 2, -2)	$(0, 1, 1,0) = X_2$

Możesz pomyśleć, że napotkasz problem, gdy wprowadzisz nowy wektor, a jeden z neuronów ma aktywację 0. W oryginalnym przykładzie znalazłeś tę sytuację, gdy aktywacja trzeciego neuronu wyjściowego wynosiła 0. Funkcja progu zapytała cię użyć tego samego wyjścia dla tego neuronu, jakie istniało we wcześniejszym cyklu czasowym. Więc wziąłeś to za 1, trzeci składnik (0, 1, 1). Ale jeśli twój wektor wejściowy jest nowym wektorem X, dla którego próbujesz znaleźć skojarzony wektor Y, to nie masz komponentu Y, na którym można by się oprzeć, gdy aktywacja okaże się równa 0. Jak więc użyć progowania funkcjonować zgodnie z opisem? Jakie masz wskazówki w tej sytuacji? Jeśli śledzisz wykorzystane wejścia i otrzymane do tej pory wyjścia, zdasz sobie sprawę, że neurony pola B (gdzie otrzymujesz wektor Y) są w pewnym stanie, co oznacza, że miały pewne wyjścia, być może z jakimś wektorem uczącym. Jeśli użyjesz tego komponentu wyjściowego jako istniejącego w poprzednim cyklu, nie będziesz mieć problemu z użyciem funkcji progowania. Jako przykład rozważmy wektor wejściowy X_3 = (1, 0, 0, 0), z którym aktywacje neuronów w Polu B byłyby (-2, 2, 0). Pierwsza składowa wektora wyjściowego to oczywiście 0, a druga to 1. Trzecia składowa jest wątpliwa. Biorąc pod uwagę ostatni wiersz tabeli, w którym Y_2 podaje stan neuronów w Polu B, możesz przyjąć 1, ostatnią składową Y_2, jako wartość uzyskaną z funkcji progowej odpowiadającej wartości aktywacji 0. Tak więc na wyjściu będzie wektorem (0, 1, 1), czyli Y_1. Ale Y_1 jest heteroskojarzone z X_1. Cóż, oznacza to, że X_3 = (1, 0, 0, 0) nie jest heteroskojarzony z żadnym wektorem X.

Przywołanie wektorów

Gdy w warstwie wejściowej zostanie przedstawiony X_1, to aktywacja w warstwie wyjściowej da (-4, 4, 2) do której stosujemy funkcję progową, która zastępuje wartość dodatnią 1, a ujemną 0. To wtedy daje nam wektor (0, 1, 1) jako wynik, który jest taki sam jak nasz Y_1. Teraz Y_1 jest przekazywane z powrotem do warstwy wejściowej przez połączenia sprzężenia zwrotnego, a aktywacja warstwy wejściowej staje się wektorem (2, –2, 2, 2), który po progowaniu daje wektor wyjściowy (1, 0, 1, 1), tak samo jak X_1. Gdy X_2 jest prezentowane w warstwie wejściowej, aktywacja w warstwie wyjściowej da (2, –2, 2) do której zostanie zastosowana funkcja progowa, która zastępuje wartość dodatnią przez 1 i wartość ujemną przez 0. To daje wektor (1, 0, 1) jako wynik, który jest taki sam jak Y_2. Teraz Y_2 jest przekazywane z powrotem do warstwy wejściowej przez połączenia zwrotne, aby uzyskać aktywację warstwy wejściowej jako (–2, 2, 2, –2), która po progowaniu daje wektor wyjściowy (0, 1, 1, 0) , czyli X_2. Dwie pary wektorów wybrane tutaj do kodowania działały dobrze, a sieć BAM z czterema neuronami w Polu A i trzema neuronami w Polu B jest gotowa do znalezienia wektora w heteroasocjacji z danym wektorem wejściowym.

Kontynuacja przykładu

Użyjmy teraz wektora X_3 = (1, 1, 1, 1). Wektor Y_3 = (0, 1, 1) jest uzyskiwany w warstwie wyjściowej. Ale następny krok, w którym przedstawiamy Y3 w kierunku wstecznym, nie daje X_3, zamiast tego daje X_1 = (1, 0, 1, 1). Mamy już X_1 powiązane z Y_1. Oznacza to, że X_3 nie jest powiązany z żadnym wektorem w przestrzeni wyjściowej. Z drugiej strony, jeśli zamiast otrzymać X1 otrzymaliśmy inny wektor X_4 i jeśli ten w operacji sprzężenia do przodu wytworzył inny wektor Y, to powtarzamy operację

sieci, dopóki na żadnym końcu nie zajdą żadne zmiany. Wtedy prawdopodobnie będziemy mieli nową parę wektorów w heteroasocjacji ustanowionej przez tę sieć BAM.

Przypadek specjalny - Uzupełnienia

Jeśli okaże się, że para (odrębnych) wzorców X i Y jest heteroskojarzona przez BAM i jeśli wprowadzisz dopełnienie X, a dopełnienie jest uzyskiwane przez zamianę zer i jedynek w X, BAM pokaże, że dopełnienie Y jest wzorzec związany z dopełnieniem X. Przykładem będzie przykładowy przebieg programu do implementacji BAM w języku C++, który przedstawiono poniżej.

Implementacja C++

W naszej implementacji C++ dyskretnej dwukierunkowej sieci asocjacyjnej pamięci tworzymy klasy dla neuronu i sieci. Inne utworzone klasy noszą nazwy exemplar, assocpair, potlpair, odpowiednio dla przykładowej pary wektorów, skojarzonej pary wektorów i potencjalnych par wektorów w celu znalezienia heteroasocjacji między nimi. Moglibyśmy stworzyć jedną klasę pairvect dla pary wektorów i wyprowadzić z niej przykład i tak dalej. Klasa sieciowa jest zadeklarowana jako klasa zaprzyjaźniona w tych innych klasach. Ale już przedstawiamy pliki nagłówkowe i źródłowe o nazwach bamntwrk.h i bamntwrk.cpp. Ponieważ ponownie wykorzystaliśmy nasz poprzedni kod z sieci Hopfield z rozdziału 4, istnieje kilka składowych danych klas, których nie użyliśmy wprost w programie. Nazywamy klasę neuronów bmneuron, aby przypomnieć nam o BAM.

Szczegóły i przebieg programu

Neuron w pierwszej warstwie to anrn, a liczba neuronów w tej warstwie to anmbr. Tablicy neuronów w drugiej warstwie nadajemy nazwę bnrn, a bnmbr oznacza rozmiar tej tablicy. Kolejność operacji w programie jest następująca:

• Prosimy użytkownika o wprowadzenie przykładowych wektorów i przekształcamy je w ich dwubiegunowe wersje. Służy do tego funkcja trnsfrm() w przykładowej klasie.

• Podajemy sieci wektor X w wersji bipolarnej w jednej przykładowej parze. Znajdujemy aktywacje elementów tablicy bnrn i otrzymujemy odpowiedni wektor wyjściowy jako wzorzec binarny. Jeśli to jest Y w przykładowej parze, sieć stworzyła pożądane skojarzenie w jednym kierunku i przechodzimy do następnego kroku. W przeciwnym razie mamy potencjalną powiązaną parę, z których jedna to X, a druga jest tym, co właśnie otrzymaliśmy jako wektor wyjściowy w przeciwległej warstwie. Mówimy potencjalną powiązaną parę, ponieważ mamy następny krok, aby potwierdzić powiązanie.

• Przeprowadzamy tablicę bnrn przez transpozycję macierzy wag i obliczamy dane wyjściowe elementów tablicy anrn. Jeśli w rezultacie otrzymamy wektor X jako tablicę anrn, znaleźliśmy powiązaną parę (X, Y). W przeciwnym razie powtarzamy dwa opisane właśnie kroki, aż znajdziemy powiązaną parę.

• Pracujemy teraz z następną parą przykładowych wektorów w taki sam sposób jak powyżej, aby znaleźć powiązaną parę.

• Powiązanym parom przypisujemy numery seryjne, oznaczane przez zmienną idn, dzięki czemu możemy wydrukować je wszystkie razem na końcu programu. Para nazywa się (X, Y), gdzie X tworzy Y poprzez macierz wag W, a Y tworzy X poprzez macierz wag, która jest transpozycją W.

• Flaga ma wartość 0 do momentu uzyskania potwierdzenia skojarzenia, gdy wartość flagi zmieni się na 1.

• Funkcje compr1 i compr2 w klasie sieci weryfikują, czy potencjalna para jest rzeczywiście powiązaną parą i ustawiają odpowiednią wartość flagi wspomnianej powyżej.

• Funkcje comput1 i comput2 w klasie sieci wykonują obliczenia w celu uzyskania aktywacji, a następnie znalezienia wektora wyjściowego we właściwych kierunkach dwukierunkowej sieci pamięci asocjacyjnej.

Przykład programu dla BAM

W naszym przebiegu ilustracyjnym przewidzieliśmy sześć neuronów w warstwie wejściowej i pięć w warstwie wyjściowej. Do kodowania użyliśmy trzech par wzorów. Użyliśmy dwóch dodatkowych wektorów wejściowych, z których jeden jest uzupełnieniem X przykładowej pary, po zakończeniu kodowania, aby zobaczyć, jakie powiązanie zostanie ustanowione w tych przypadkach lub jakie wywołanie zostanie wykonane przez sieć BAM.

Plik nagłówka

Jak oczekiwano, dopełnienie Y wzorca jest powiązane z dopełnieniem X tej pary. Jednak gdy prezentowany jest drugi wektor wejściowy, zostaje znaleziona nowa para skojarzonych wektorów. Po przedstawieniu kodu wyświetlamy również dane wyjściowe komputera.

Listing 1 bamntwrk.h

```
//bamntwrk.h V. Rao, H. Rao

//Header file for BAM network program

#include <iostream.h>

#include <math.h>

#include <stdlib.h>

#define MXSIZ 10 // determines the maximum size of the network

class bmneuron

{

protected:

int nnbr;

int inn,outn;

int output;

int activation;

int outwt[MXSIZ];

char *name;

friend class network;

public:

bmneuron() { };
```

```
void getnrn(int,int,int,char *);
};
class exemplar
{
protected:
int xdim,ydim;
int v1[MXSIZ],v2[MXSIZ];
int u1[MXSIZ],u2[MXSIZ];
friend class network;
friend class mtrx;
public:
exemplar() { };
void getexmplr(int,int,int *,int *);
void prexmplr();
void trnsfrm();
void prtrnsfrm();
};
class asscpair
{
protected:
int xdim,ydim,idn;
int v1[MXSIZ],v2[MXSIZ];
friend class network;
public:
asscpair() { };
void getasscpair(int,int,int);
void prasscpair();
};
class potlpair
{
protected:
```

```
int xdim,ydim;
int v1[MXSIZ],v2[MXSIZ];
friend class network;
public:
potlpair() { };
void getpotlpair(int,int);
void prpotlpair();
};
class network
{
public:
int anmbr,bnmbr,flag,nexmplr,nasspr,ninpt;
bmneuron (anrn)[MXSIZ],(bnrn)[MXSIZ];
exemplar (e)[MXSIZ];
asscpair (as)[MXSIZ];
potlpair (pp)[MXSIZ];
int outs1[MXSIZ],outs2[MXSIZ];
int mtrx1[MXSIZ][MXSIZ],mtrx2[MXSIZ][MXSIZ];
network() { };
void getnwk(int,int,int,int [][6],int [][5]);
void compr1(int,int);
void compr2(int,int);
void prwts();
void iterate();
void findassc(int *);
void asgninpt(int *);
void asgnvect(int,int *,int *);
void comput1();
void comput2();
void prstatus();
};
```

Plik źródłowy

Plik źródłowy programu jest przedstawiony w następujący sposób

Listing 8.2 bamntwrk.cpp

```
//bamntwrk.cpp V. Rao, H. Rao
//Source file for BAM network program
#include “bamntwrk.h”
void bmneuron::getnrn(int m1,int m2,int m3,char *y)
{
int i;
name = y;
nnbr = m1;
outn = m2;
inn = m3;
for(i=0;i<outn;++i){
outwt[i] = 0 ;
}
output = 0;
activation = 0;
}
void exemplar::getexmplr(int k,int l,int *b1,int *b2)
{
int i2;
xdim = k;
ydim = l;
for(i2=0;i2<xdim;++i2){
v1[i2] = b1[i2]; }
for(i2=0;i2<ydim;++i2){
v2[i2] = b2[i2]; }
}
void exemplar::prexmplr()
{
```

```
int i;
cout<<"\nX vector you gave is:\n";
for(i=0;i<xdim;++i){
cout<<v1[i]<<" ";}
cout<<"\nY vector you gave is:\n";
for(i=0;i<ydim;++i){
cout<<v2[i]<<" ";}
cout<<"\n";
}
void exemplar::trnsfrm()
{
int i;
for(i=0;i<xdim;++i){
u1[i] = 2*v1[i] -1;}
for(i=0;i<ydim;++i){
u2[i] = 2*v2[i] - 1;}
}
void exemplar::prtrnsfrm()
{
int i;
cout<<"\nbipolar version of X vector you gave is:\n";
for(i=0;i<xdim;++i){
cout<<u1[i]<<" ";}
cout<<"\nbipolar version of Y vector you gave is:\n";
for(i=0;i<ydim;++i){
cout<<u2[i]<<" ";}
cout<<"\n";
}
void asscpair::getasscpair(int i,int j,int k)
{
idn = i;
```

```
xdim = j;
ydim = k;
}
void asscpair::prasscpair()
{
int i;
cout<<"\nX vector in the associated pair no. "<<idn<<" is:\n";
for(i=0;i<xdim;++i){
cout<<v1[i]<<" ";}
cout<<"\nY vector in the associated pair no. "<<idn<<" is:\n";
for(i=0;i<ydim;++i){
cout<<v2[i]<<" ";}
cout<<"\n";
}
void potlpair::getpotlpair(int k,int j)
{
xdim = k;
ydim = j;
}
void potlpair::prpotlpair()
{
int i;
cout<<"\nX vector in possible associated pair is:\n";
for(i=0;i<xdim;++i){
cout<<v1[i]<<" ";}
cout<<"\nY vector in possible associated pair is:\n";
for(i=0;i<ydim;++i){
cout<<v2[i]<<" ";}
cout<<"\n";
}
void network::getnwk(int k,int l,int k1,int b1[][6],int
```

```
b2[][5])
{
anmbr = k;
bnmbr = l;
nexmplr = k1;
nasspr = 0;
ninpt = 0;
int i,j,i2;
flag =0;
char *y1=”ANEURON”, *y2=”BNEURON” ;
for(i=0;i<nexmplr;++i){
e[i].getexmplr(anmbr,bnmbr,b1[i],b2[i]);
e[i].prexmplr();
e[i].trnsfrm();
e[i].prtrnsfrm();
}
for(i=0;i<anmbr;++i){
anrn[i].bmneuron::getnrn(i,bnmbr,0,y1);}
for(i=0;i<bnmbr;++i){
bnrn[i].bmneuron::getnrn(i,0,anmbr,y2);}
for(i=0;i<anmbr;++i){
for(j=0;j<bnmbr;++j){
mtrx1[i][j] = 0;
for(i2=0;i2<nexmplr;++i2){
mtrx1[i][j] += e[i2].u1[i]*e[i2].u2[j];}
mtrx2[j][i] = mtrx1[i][j];
anrn[i].outwt[j] = mtrx1[i][j];
bnrn[j].outwt[i] = mtrx2[j][i];
}
}
prwts();
```

```
cout<<"\n";
}
void network::asgninpt(int *b)
{
int i;
cout<<"\n";
for(i=0;i<anmbr;++i){
anrn[i].output = b[i];
outs1[i] = b[i];
}
}
void network::compr1(int j,int k)
{
int i;
for(i=0;i<anmbr;++i){
if(pp[j].v1[i] != pp[k].v1[i]) flag = 1;
break;
}
}
void network::compr2(int j,int k)
{
int i;
for(i=0;i<anmbr;++i){
if(pp[j].v2[i] != pp[k].v2[i]) flag = 1;
break;}
}
void network::comput1()
{
int j;
for(j=0;j<bnmbr;++j){
int ii1;
```

```
int c1 =0,d1;
cout<<"\n";
for(ii1=0;ii1<anmbr;++ii1){
d1 = outs1[ii1] * mtrx1[ii1][j];
c1 += d1;
}
bnrn[j].activation = c1;
cout<<"\n output layer neuron "<<j<<" activation is"
<<c1<<"\n";
if(bnrn[j].activation <0) {
bnrn[j].output = 0;
outs2[j] = 0;}
else
if(bnrn[j].activation>0) {
bnrn[j].output = 1;
outs2[j] = 1;}
else
{cout<<"\n A 0 is obtained, use previous output
value \n";
if(ninpt<=nexmplr){
bnrn[j].output = e[ninpt-1].v2[j];}
else
{ bnrn[j].output = pp[0].v2[j];}
outs2[j] = bnrn[j].output; }
cout<<"\n output layer neuron "<<j<<" output is"
<<bnrn[j].output<<"\n";
}
}
void network::comput2()
{
int i;
```

```
for(i=0;i<anmbr;++i){
int ii1;
int c1=0;
for(ii1=0;ii1<bnmbr;++ii1){
c1 += outs2[ii1] * mtrx2[ii1][i]; }
anrn[i].activation = c1;
cout<<"\ninput layer neuron "<<i<<"activation is "
<<c1<<"\n";
if(anrn[i].activation <0 ){
anrn[i].output = 0;
outs1[i] = 0;}
else
if(anrn[i].activation >0 ) {
anrn[i].output = 1;
outs1[i] = 1;
}
else
{ cout<<"\n A 0 is obtained, use previous value if available\n";
if(ninpt<=nexmplr){
anrn[i].output = e[ninpt-1].v1[i];}
else
{anrn[i].output = pp[0].v1[i];}
outs1[i] = anrn[i].output;}
cout<<"\n input layer neuron
"<<i<<" output is "
<<anrn[i].output<<"\n";
}
}
void network::asgnvect(int j1,int *b1,int *b2)
{
int j2;
```

```
for(j2=0;j2<j1;++j2){
b2[j2] = b1[j2];}
}
void network::prwts()
{
int i3,i4;
cout<<"\n weights— input layer to output layer: \n\n";
for(i3=0;i3<anmbr;++i3){
for(i4=0;i4<bnmbr;++i4){
cout<<anrn[i3].outwt[i4]<<" ";}
cout<<"\n"; }
cout<<"\n";
cout<<"\nweights— output layer to input layer: \n\n";
for(i3=0;i3<bnmbr;++i3){
for(i4=0;i4<anmbr;++i4){
cout<<bnrn[i3].outwt[i4]<<" ";}
cout<<"\n"; }
cout<<"\n";
}
void network::iterate()
{
int i1;
for(i1=0;i1<nexmplr;++i1){
findassc(e[i1].v1);
}
}
void network::findassc(int *b)
{
int j;
flag = 0;
asgninpt(b);
```

```
ninpt ++;
cout<<”\nInput vector is:\n” ;
for(j=0;j<6;++j){
cout<<b[j]<<” “;}
cout<<”\n”;
pp[0].getpotlpair(anmbr,bnmbr);
asgnvect(anmbr,outs1,pp[0].v1);
comput1();
if(flag>=0){
asgnvect(bnmbr,outs2,pp[0].v2);
cout<<”\n”;
pp[0].prpotlpair();
cout<<”\n”;
comput2(); }
cout<<anrn[i3].outwt[i4]<<” “;}
cout<<”\n”; }
cout<<”\n”;
cout<<”\nweights— output layer to input layer: \n\n”;
for(i3=0;i3<bnmbr;++i3){
for(i4=0;i4<anmbr;++i4){
cout<<bnrn[i3].outwt[i4]<<” “;}
cout<<”\n”; }
cout<<”\n”;
}
void network::iterate()
{
int i1;
for(i1=0;i1<nexmplr;++i1){
findassc(e[i1].v1);
}
}
```

```
void network::findassc(int *b)
{
int j;
flag = 0;
asgninpt(b);
ninpt ++;
cout<<"\nInput vector is:\n" ;
for(j=0;j<6;++j){
cout<<b[j]<<" ";}
cout<<"\n";
pp[0].getpotlpair(anmbr,bnmbr);
asgnvect(anmbr,outs1,pp[0].v1);
comput1();
if(flag>=0){
asgnvect(bnmbr,outs2,pp[0].v2);
cout<<"\n";
pp[0].prpotlpair();
cout<<"\n";
comput2(); }
for(j=1;j<MXSIZ;++j){
pp[j].getpotlpair(anmbr,bnmbr);
asgnvect(anmbr,outs1,pp[j].v1);
comput1();
asgnvect(bnmbr,outs2,pp[j].v2);
pp[j].prpotlpair();
cout<<"\n";
compr1(j,j-1);
compr2(j,j-1);
if(flag == 0) {
int j2;
nasspr += 1;
```

```
j2 = nasspr;
as[j2].getasscpair(j2,anmbr,bnmbr);
asgnvect(anmbr,pp[j].v1,as[j2].v1);
asgnvect(bnmbr,pp[j].v2,as[j2].v2);
cout<<"\nPATTERNS ASSOCIATED:\n";
as[j2].prasscpair();
j = MXSIZ ;
}
else
if(flag == 1)
{
flag = 0;
comput1();
}
}
}
void network::prstatus()
{
int j;
cout<<"\nTHE FOLLOWING ASSOCIATED PAIRS WERE FOUND BY BAM\n\n";
for(j=1;j<=nasspr;++j){
as[j].prasscpair();
cout<<"\n";}
}
void main()
{
int ar = 6, br = 5, nex = 3;
int inptv[][6]={1,0,1,0,1,0,1,1,1,0,0,0,0,1,1,0,0,0,0,1,0,1,0,1,
1,1,1,1,1,1};
int outv[][5]={1,1,0,0,1,0,1,0,1,1,1,0,0,1,0};
cout<<"\n\nTHIS PROGRAM IS FOR A BIDIRECTIONAL ASSOCIATIVE MEMORY NETWORK.\n";
```

```
cout<<" THE NETWORK ISSET UP FOR ILLUSTRATION WITH "<<ar<<

" INPUT NEURONS, AND "<<br;

cout<<" OUTPUT NEURONS.\n"<<nex

<<" exemplars are used to encode \n";

static network bamn;

bamn.getnwk(ar,br,nex,inptv,outv) ;

bamn.iterate();

bamn.findassc(inptv[3]);

bamn.findassc(inptv[4]);

bamn.prstatus();

}
```

Wyjście programu

Wyniki z przykładowego uruchomienia programu są wymienione w następnej kolejności. Zauważysz, że przewidzieliśmy wiele informacji, które mają być wyprowadzane podczas wykonywania programu. Czwarty wektor, który wprowadzamy, nie pochodzi z wzorca, ale jest uzupełnieniem X pierwszej pary wzorców. Sieć stwierdziła, że uzupełnienie Y tego przykładu jest powiązane z tym wejściem. Piąty wektor, który wprowadzamy, to (1, 1, 1, 1, 1, 1). Ale BAM przypomniał w tym przypadku parę dopełniaczy dla trzecie przykładowej pary, czyli X = (1, 0, 0, 1, 1, 1) i Y = (0, 1, 1, 0, 1). Zauważasz, że odległość Hamminga tego wzoru wejściowego od X trzech przykładów wynosi odpowiednio 3, 3 i 4. Odległość Hamminga od dopełnienia X pierwszej pary wzorców również wynosi 3. Ale odległość Hamminga (1, 1, 1, 1) od X dopełnienia X trzeciej pary wzorców wynosi tylko 2. Byłoby pouczające, gdybyś użył wzorca wejściowego (1, 1, 1, 1, 1, 0), aby zobaczyć, do jakiej skojarzonej pary prowadzi. Teraz wyjście:

```
THIS PROGRAM IS FOR A BIDIRECTIONAL ASSOCIATIVE

MEMORY NETWORK.

THE NETWORK IS SET UP FOR ILLUSTRATION WITH SIX INPUT

NEURONS AND FIVE OUTPUT NEURONS.

Three exemplars are used to encode

X vector you gave is:

1 0 1 0 1 0

Y vector you gave is:

1 1 0 0 1

bipolar version of X vector you gave is:

1 -1 1 -1 1 -1

bipolar version of Y vector you gave is:
```

1 1 -1 -1 1

X vector you gave is:

1 1 1 0 0 0

Y vector you gave is:

0 1 0 1 1

bipolar version of X vector you gave is:

1 1 1 -1 -1 -1

bipolar version of Y vector you gave is:

-1 1 -1 1 1

X vector you gave is:

0 1 1 0 0 0

Y vector you gave is:

1 0 0 1 0

bipolar version of X vector you gave is:

-1 1 1 -1 -1 -1

bipolar version of Y vector you gave is:

1 -1 -1 1 -1

weights— input layer to output layer:

-1 3 -1 -1 3

-1 1 -1 3 -1

1 1 -3 1 1

-1 -1 3 -1 -1

1 1 1 -3 1

-1 -1 3 -1 -1

weights— output layer to input layer:

-1 -1 1 -1 1 -1

3 -1 1 -1 1 -1

-1 -1 -3 3 1 3

-1 3 1 -1 -3 -1

3 -1 1 -1 1 -1

Input vector is:

1 0 1 0 1 0

output layer neuron 0 activation is 1

output layer neuron 0 output is 1

output layer neuron 1 activation is 5

output layer neuron 1 output is 1

output layer neuron 2 activation is -3

output layer neuron 2 output is 0

output layer neuron 3 activation is -3

output layer neuron 3 output is 0

output layer neuron 4 activation is 5

output layer neuron 4 output is 1

X vector in possible associated pair is:

1 0 1 0 1 0

Y vector in possible associated pair is:

1 1 0 0 1

input layer neuron 0 activation is 5

input layer neuron 0 output is 1

input layer neuron 1 activation is -3

input layer neuron 1 output is 0

input layer neuron 2 activation is 3

input layer neuron 2 output is 1

input layer neuron 3 activation is -3

input layer neuron 3 output is 0

input layer neuron 4 activation is 3

input layer neuron 4 output is 1

input layer neuron 5 activation is -3

input layer neuron 5 output is 0

output layer neuron 0 activation is 1

output layer neuron 0 output is 1

output layer neuron 1 activation is 5

output layer neuron 1 output is 1

output layer neuron 2 activation is -3

output layer neuron 2 output is 0

output layer neuron 3 activation is -3

output layer neuron 3 output is 0

output layer neuron 4 activation is 5

output layer neuron 4 output is 1

X vector in possible associated pair is:

1 0 1 0 1 0

Y vector in possible associated pair is:

1 1 0 0 1

PATTERNS ASSOCIATED:

X vector in the associated pair no. 1 is:

1 0 1 0 1 0

Y vector in the associated pair no. 1 is:

1 1 0 0 1

Input vector is:

1 1 1 0 0 0

// We get here more of the detailed output as in the previous case. We will simply

not present it here.

PATTERNS ASSOCIATED:

X vector in the associated pair no. 1 is:

1 1 1 0 0 0

Y vector in the associated pair no. 1 is:

0 1 0 1 1

Input vector is:

0 1 1 0 0 0

output layer neuron 0 activation is 0

A 0 is obtained, use previous output value

output layer neuron 0 output is 1

output layer neuron 1 activation is 0

A 0 is obtained, use previous output value

output layer neuron 1 output is 0

output layer neuron 2 activation is -4

output layer neuron 2 output is 0

output layer neuron 3 activation is 4

output layer neuron 3 output is 1

output layer neuron 4 activation is 0

A 0 is obtained, use previous output value

output layer neuron 4 output is 0

X vector in possible associated pair is:

0 1 1 0 0 0

Y vector in possible associated pair is:

1 0 0 1 0

// We get here more of the detailed output as in the previous case. We will simply not present it here.

PATTERNS ASSOCIATED:

X vector in the associated pair no. 1 is:

0 1 1 0 0 0

Y vector in the associated pair no. 1 is:

1 0 0 1 0

Input vector is:

0 1 0 1 0 1

// We get here more of the detailed output as in the previous case. We will simply not present it here.

X vector in possible associated pair is:

0 1 0 1 0 1

Y vector in possible associated pair is:

0 0 1 1 0

// We get here more of the detailed output as in the previous case. We will simply not present it here.

X vector in possible associated pair is:

0 1 0 1 0 1

Y vector in possible associated pair is:

0 0 1 1 0

PATTERNS ASSOCIATED:

X vector in the associated pair no. 1 is:

0 1 0 1 0 1

Y vector in the associated pair no. 1 is:

0 0 1 1 0

Input vector is:

1 1 1 1 1 1

output layer neuron 0 activation is -2

output layer neuron 0 output is 0

output layer neuron 1 activation is 2

output layer neuron 1 output is 1

output layer neuron 2 activation is 2

output layer neuron 2 output is 1

output layer neuron 3 activation is -2

output layer neuron 3 output is 0

output layer neuron 4 activation is 2

output layer neuron 4 output is 1

X vector in possible associated pair is:

1 1 1 1 1 1

Y vector in possible associated pair is:

0 1 1 0 1

input layer neuron 0 activation is 5

input layer neuron 0 output is 1

input layer neuron 1 activation is -3

input layer neuron 1 output is 0

input layer neuron 2 activation is -1

input layer neuron 2 output is 0

input layer neuron 3 activation is 1

input layer neuron 3 output is 1

input layer neuron 4 activation is 3

input layer neuron 4 output is 1

input layer neuron 5 activation is 1

input layer neuron 5 output is 1

output layer neuron 0 activation is -2

output layer neuron 0 output is 0

output layer neuron 1 activation is 2

output layer neuron 1 output is 1

output layer neuron 2 activation is 6

output layer neuron 2 output is 1

output layer neuron 3 activation is -6

output layer neuron 3 output is 0

output layer neuron 4 activation is 2

output layer neuron 4 output is 1

X vector in possible associated pair is:

1 0 0 1 1 1

Y vector in possible associated pair is:

0 1 1 0 1

PATTERNS ASSOCIATED:

X vector in the associated pair no. 1 is:

1 0 0 1 1 1

Y vector in the associated pair no. 1 is:

0 1 1 0 1

THE FOLLOWING ASSOCIATED PAIRS WERE FOUND BY BAM

X vector in the associated pair no. 1 is: //first exemplar pair

1 0 1 0 1 0

Y vector in the associated pair no. 1 is:

1 1 0 0 1

X vector in the associated pair no. 2 is: //second exemplar pair

1 1 1 0 0 0

Y vector in the associated pair no. 2 is:

0 1 0 1 1

X vector in the associated pair no. 3 is: //third exemplar pair

0 1 1 0 0 0

Y vector in the associated pair no. 3 is:

1 0 0 1 0

X vector in the associated pair no. 4 is: //complement of X of the

0 1 0 1 0 1 first exemplar pair

Y vector in the associated pair no. 4 is: //complement of Y of the

0 0 1 1 0 first exemplar pair

X vector in the associated pair no. 5 is: //input was X = (1, 1, 1,

1 0 0 1 1 1 1, 1) but result was complement

of third exemplar pair

Y vector in the associated pair no. 5 is: with X of which Hamming

0 1 1 0 1 distance is the least.

Dodatkowe problemy

Jeśli chcesz używać wektorów ze składnikami liczb rzeczywistych, zamiast liczb binarnych, możesz to zrobić. Twój model jest wtedy nazywany ciągłą dwukierunkową pamięcią asocjacyjną. Macierz W i jej transpozycja są używane do wag połączeń. Jednak macierz W nie jest sformułowana w sposób opisany do tej pory. Macierz jest arbitralnie wybrana i utrzymywana na stałym poziomie. Funkcja progowa jest również wybierana jako funkcja ciągła, w przeciwieństwie do tej używanej wcześniej. Zmiany w aktywacji neuronów podczas treningu są zgodne z rozszerzeniem paradygmatu Cohena-Grossberga. Michael A. Cohen i Stephen Grossberg opracowali swój model autoasocjacji z symetryczną macierzą wag. Stabilność zapewnia twierdzenie Cohena-Grossberga; nie ma nauki. Jeśli jednak chcesz, aby sieć uczyła się nowych par wzorców, modyfikując wagi tak, aby znaleźć powiązania między nowymi parami, projektujesz coś, co nazywa się adaptacyjną dwukierunkową pamięcią skojarzeniową (ABAM).

Prawo, które rządzi zmianami wagi, określa rodzaj ABAM, a mianowicie konkurencyjny ABAM, różnicowy hebbański ABAM lub różnicowy konkurencyjny ABAM. W przeciwieństwie do modelu ABAM, który jest typu addytywnego, w pozostałych modelach stosuje się niektóre iloczyny wyjść z dwóch warstw lub pochodne funkcji progowych. Poniżej przedstawiamy krótki opis modelu, który jest odmianą BAM. Nazywa się UBBAM (Unipolar Binary Bidirectional Associative Memory).

Jednobiegunowa binarna dwukierunkowa pamięć asocjacyjna

T. C. B. Yu i RJ Mears opisują projekt jednobiegunowej binarnej dwukierunkowej pamięci asocjacyjnej i jej implementację za pomocą inteligentnego zaawansowanego modulatora światła przestrzennego (SASLM). Urządzenie SASLM to ferroelektryczny ciekłokrystaliczny przestrzenny modulator światła. Jest napędzany przez krzemową płytę montażową CMOS. Używamy ich artykułu do przedstawienia niektórych cech jednobiegunowej binarnej dwukierunkowej pamięci asocjacyjnej i ignorujemy jej implementację sprzętową. Przypomnij sobie procedurę, za pomocą której określasz macierz wag W dla sieci BAM, opisaną na poprzednich stronach. W pierwszym kroku konwertujesz każdy wektor w

przykładowych parach na jego wersję dwubiegunową. Jeśli X i Y są przykładową parą (w wersjach bipolarnych), bierzesz iloczyn XT Y i dodajesz go do podobnych produktów z innych przykładowych par, aby otrzymać macierz wag W. Niektóre elementy macierzy W mogą być liczbami ujemnymi . W kontekście jednobiegunowym nie masz jednocześnie wartości ujemnych i dodatnich. Dozwolony jest tylko jeden z nich. Załóżmy, że nie chcesz żadnych liczb ujemnych; wtedy jednym ze sposobów naprawienia sytuacji jest dodanie wystarczająco dużej stałej do każdego elementu macierzy. Nie możesz dodać tylko do liczb ujemnych, które pojawiają się w macierzy. Spójrzmy na przykład. Załóżmy, że wybierasz dwie pary wektorów jako możliwe przykłady. Pozwól im być,

$X_1 = (1, 0, 0, 1)$, Y1= (0, 1, 1)

oraz

$X_2 = (0, 1, 1, 0)$, Y2 = (1, 0, 1)

Zamieniasz je w składowe dwubiegunowe i otrzymujesz odpowiednio (1, –1, –1, 1), (-1, 1, 1), (-1, 1, 1, –1) i (1, –1 , 1). Obliczenie W dla BAM przeprowadzono w następujący sposób.

$$W = \begin{matrix}1\\-1\\-1\\1\end{matrix}\begin{bmatrix}-1 & 1 & 1\end{bmatrix} + \begin{matrix}-1\\1\\-1\\1\end{matrix}\begin{bmatrix}1 & -1 & 1\end{bmatrix} = \begin{matrix}-1 & 1 & 1\\1 & -1 & -1\\1 & -1 & -1\\-1 & 1 & 1\end{matrix} + \begin{matrix}-1 & 1 & -1\\1 & -1 & 1\\1 & -1 & 1\\-1 & 1 & -1\end{matrix} = \begin{matrix}-2 & 2 & 0\\2 & -2 & 0\\2 & -2 & 0\\-2 & 2 & 0\end{matrix}$$

i

$$W^T = \begin{matrix}-2 & 2 & 2 & -2\\2 & -2 & -2 & 2\\0 & 0 & 0 & 0\end{matrix}$$

Widzisz kilka liczb ujemnych w macierzy W. Jeśli dodasz stałą m, m = 2, do wszystkich elementów macierzy, otrzymasz następującą macierz, oznaczoną przez W~.

$$W^{\sim} = \begin{matrix}0 & 4 & 2\\4 & 0 & 2\\4 & 0 & 2\\0 & 4 & 2\end{matrix}$$

Potrzebujesz również modyfikacji funkcji progowania. Wcześniej miałeś następującą funkcję.

$$b_j|_{t+1} = \begin{cases}1 & \text{if } y_j > 0\\ b_j|_t & \text{if } y_j = 0\\ 0 & \text{if } y_j < 0\end{cases} \quad \text{and} \quad a_i|_{t+1} = \begin{cases}1 & \text{if } x_i > 0\\ a_i|_t & \text{if } x_i = 0\\ 0 & \text{if } x_i < 0\end{cases}$$

Teraz musisz zmienić prawe strony warunków z 0 na iloczyn mi sumy wejść neuronów, czyli na m razy sumę wejść. Dla zwięzłości użyjmy S_i jako sumy danych wejściowych. Nie jest to stała, a jej wartość zależy od tego, jakie wartości uzyskasz dla wejść neuronowych w dowolnym kierunku. Wtedy funkcję progową można podać w następujący sposób:

$$b_j|_{t+1} = \begin{cases} 1 & \text{if } y_j > m\,S_i \\ b_j|_t & \text{if } y_j = m\,S_i \\ 0 & \text{if } y_j < m\,S_i \end{cases} \quad \text{and} \quad a_i|_{t+1} = \begin{cases} 1 & \text{if } x_i > m\,S_i \\ a_i|_t & \text{if } x_i = m\,S_i \\ 0 & \text{if } x_i < m\,S_i \end{cases}$$

Na przykład wektor X_1 = (1, 0, 0, 1) ma wektor aktywności (0, 8, 4), a odpowiadający mu wektor wyjściowy to (0, 1, 1), czyli Y_1 = (0, 1 , 1). Wartość Si wynosi 2 = 1 + 0 + 0 + 1. Zatem mS_i = 4. Pierwsza składowa wektora aktywacji wynosi 0, która jest mniejsza niż 4, a więc pierwsza składowa wektora wyjściowego wynosi 0. drugi składnik wektora aktywacji to 8, który jest większy niż 4, a więc drugi składnik wektora wyjściowego to 1. Ostatni składnik wektora aktywności to 4, co jest równe mS_i, a więc używasz trzeciego składnika wektora Y_1 = (0, 1, 1), czyli 1, jako trzecia składowa wyjścia.

Przykład Yu i Mearsa składa się z ośmiu 8-komponentowych wektorów w obu kierunkach modelu Dwukierunkowej Pamięci Asocjacyjnej. Możesz wziąć te wektory, tak jak one, do zdefiniowania wartości pikseli w siatce 8x8 pikseli. Udało się osiągnąć powiązanie dwóch wzorców przestrzennych. Jeśli chodzi o liczby binarne, które pokazują wartości pikseli, wzory pokazano na rysunku.

Nazwijcie je Wzorem I i Wzorem II. Zamiast Wzorca I, użyli jego zniekształconej formy, jak pokazano na rysunku.

W sieci nie było problemu ze znalezieniem skojarzonej pary (Wzorzec I, Wzorzec II). Na rysunku powyżej uszkodzona wersja Wzorca I różni się od Wzorca I, w 10 z 64 miejsc, zepsucie 15,6%. Ten procent zepsucia jest podawany jako granica, poniżej której, według Yu i Mearsa, uzyskuje się przywoływanie heteroasocjacyjne. Dzięki temu modelowi możliwa jest do pewnego stopnia eliminacja hałasu. Jak widzieliśmy w przypadku pamięci Hopfielda, zastosowaniem pamięci skojarzeniowej jest uzupełnianie wzorca, gdzie przedstawiana jest uszkodzona wersja wzorca i przywołuje się prawdziwy wzorzec. To jest autoaskojarzenie. W przypadku BAM masz wywołanie heteroskojarzeniowe z uszkodzonymi danymi wejściowymi.

Posumowanie

Zaprezentowane są dwukierunkowe wspomnienia skojarzeniowe. Rozwój tych wspomnień to w dużej mierze zasługa Kosko. Dzielą one z adaptacyjną teorią rezonansu cechę rezonansu między dwiema warstwami w sieci. Skojarzeniowa dwukierunkowa sieć pamięci (BAM) znajduje heteroasocjację między wzorcami binarnymi, które są konwertowane na wartości dwubiegunowe w celu określenia macierzy wagi połączenia. Mimo że istnieją połączenia w obu kierunkach między neuronami w dwóch warstwach, zasadniczo w grę wchodzi tylko jedna macierz wag. Użyj transpozycji tej macierzy wag dla połączeń w przeciwnym kierunku. Kiedy jedno wejście na jednym końcu prowadzi do jakiegoś wyjścia

na drugim końcu, co z kolei prowadzi do wyjścia, które jest takie samo jak poprzednie, osiągany jest rezonans i zostaje znaleziona powiązana para. Ciągła dwukierunkowa pamięć asocjacyjna rozszerza model binarny na przypadek ciągły. Adaptacyjne dwukierunkowe wspomnienia o różnych cechach są wynikiem włączenia różnych paradygmatów uczenia się. Jednobiegunowa wersja binarna BAM jest również prezentowana jako aplikacja BAM do uzupełniania wzorca.