FAM: Rozmyta pamięć skojarzeniowa

Wstęp

Tu poznamy zbiory rozmyte i ich elementy, zarówno dla wejścia, jak i wyjścia asocjacyjnej sieci neuronowej. Każdy element zbioru rozmytego ma stopień przynależności do zbioru. O ile ten stopień przynależności nie wynosi 1, element nie należy do zbioru (w sensie elementów zwykłego zbioru należących do zbioru). W sieci neuronowej systemu rozmytego wszystkie wejścia, wyjścia i wagi połączeń należą do przestrzeni, które je definiują. Macierz wag będzie macierzą rozmytą, a aktywacje neuronów w takiej sieci muszą być określone regułami logiki rozmytej i operacji na zbiorach rozmytych. System ekspercki używa tak zwanych ścisłych reguł i stosuje je sekwencyjnie. Zaletą rzutowania tego samego problemu w systemie rozmytym jest to, że reguły, z którymi pracujesz, nie muszą być ostre, a przetwarzanie odbywa się równolegle. To, co mogą określić systemy rozmyte, to asocjacja rozmyta. Powiązania te mogą być modyfikowane, a leżące u ich podstaw zjawiska lepiej rozumiane w miarę zdobywania doświadczenia. To jeden z powodów ich rosnącej popularności w aplikacjach. Kiedy próbujemy powiązać dwie rzeczy metodą prób i błędów, domyślnie i intuicyjnie ustanawiamy skojarzenie, które jest stopniowo modyfikowane i być może w pewnym sensie ulepszane. W takim ćwiczeniu może występować kilka zmiennych rozmytych. To, że na początku nie mieliśmy pełnej wiedzy, nie jest przeszkodą; istnieje pewna różnica w korzystaniu z prawdopodobieństw i logiki rozmytej. Stopień przynależności przypisany do elementu zbioru nie musi być tak stanowczy jak przypisanie prawdopodobieństwa. Stopień przynależności jest, podobnie jak prawdopodobieństwo, liczbą rzeczywistą z zakresu od 0 do 1. Im bliżej 1, tym mniej niejednoznaczna jest przynależność elementu w danym zbiorze. Załóżmy, że masz zbiór, który może zawierać lub nie trzy elementy, powiedzmy a, b i c. Wtedy jego reprezentacją w zbiorze rozmytym byłaby uporządkowana trójka (m_a, m_b, m_c), która nazywana jest wektorem dopasowania, a jej składowe nazywane są wartościami dopasowania. Na przykład trójka (0,5, 0,5, 0,5) pokazuje, że każdy z a, b i c ma członkostwo równe tylko połowie. Ta trójka sama w sobie opisze zbiór rozmyty. Można go również traktować jako punkt w przestrzeni trójwymiarowej. Żaden z takich punktów nie będzie znajdował się poza kostką jednostki. Gdy liczba elementów jest większa, odpowiednie punkty będą znajdować się na lub wewnątrz hipersześcianu jednostki. Warto zauważyć, że ten rozmyty zbiór, podany przez trójkę (0.5, 0.5, 0.5), jest jego własnym uzupełnieniem, co nie zdarza się w zwykłych zbiorach. Uzupełnieniem jest zestaw, który pokazuje stopnie nieprzynależności. Wysokość zbioru rozmytego to maksimum jego wartości dopasowania, a o zbiorze rozmytym mówi się, że jest normalny, jeśli jego wysokość wynosi 1. Zbiór rozmyty z wektorem dopasowania (0,3, 0,7, 0,4) ma wysokość 0,7 i jest nie normalny zestaw rozmyty. Jednak wprowadzając dodatkowy fikcyjny komponent o wartości dopasowania 1, możemy rozszerzyć go do normalnego zestawu rozmytego. Celowość normalności zbioru rozmytego stanie się oczywista, gdy będziemy mówić o przywoływaniu w rozmytych wspomnieniach skojarzeniowych. Relacja podzbiorów jest również inna dla zbiorów rozmytych niż sposób jej definiowania dla zbiorów zwykłych. Na przykład, jeśli masz zbiór rozmyty podany przez trójkę (0.3, 0.7, 0.4), to dowolny zbiór rozmyty z trójką (a, b, c) taki, że $a \leq 0,3$, $b \leq 0,7$ i $c \leq 0,4$, jest jego podzbiorem rozmytym. Na przykład zbiór rozmyty podany przez trójkę (0,1, 0,7, 0) jest podzbiorem zbioru rozmytego (0,3, 0,7, 0,4).

Stowarzyszenie

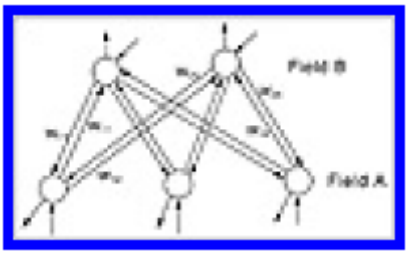
Rozważmy dwa rozmyte zestawy, jeden być może odwołujący się do popularności (lub zainteresowania) wystawą, a drugi do ceny wstępu. Popularność może być bardzo wysoka, wysoka, uczciwa, niska lub bardzo niska. W związku z tym wektor dopasowania będzie miał pięć komponentów.

Cena

Wstęp może być wysoki, skromny lub niski. Jego wektor dopasowania składa się z trzech elementów. Rozmyty system pamięci asocjacyjnej ma wtedy skojarzenie (popularność, cena), a para zbiorów rozmytych koduje to skojarzenie. W kolejnych sekcjach opisujemy proces kodowania i wycofania. Po zakończeniu kodowania skojarzenia między podzbiorami pierwszego zestawu rozmytego i podzbiorami drugiego zestawu rozmytego są również ustanawiane przez ten sam rozmyty system pamięci asocjacyjnej.

Sieć neuronowa FAM

Sieć neuronowa dla rozmytej pamięci asocjacyjnej ma warstwę wejściową i wyjściową z pełnymi połączeniami w obu kierunkach, tak jak w sieci neuronowej BAM. Rysunek 1 przedstawia układ. Aby uniknąć bałaganu, rysunek przedstawia sieć z trzema neuronami w polu A i dwoma neuronami w polu B i tylko niektórymi z ósemek połączeń. Ogólny przypadek jest analogiczny.



Kodowanie

Kodowanie dla rozmytych systemów pamięci asocjacyjnej jest koncepcyjnie podobne do procesu kodowania dwukierunkowych pamięci asocjacyjnych, z pewnymi różnicami w zaangażowanych operacjach. W kodowaniu BAM bipolarne wersje wektorów binarnych są używane tylko dla kodowania. Macierze typu $X_i^T Y_i$ są sumowane w celu uzyskania macierzy wag połączeń. Są dwie podstawowe operacje na elementach tych wektorów. Są mnożeniem i dodawaniem produktów. Nie ma konwersji wartości dopasowania przed kodowaniem przez zestawy rozmyte. Mnożenie elementów zostaje zastąpione operacją brania minimum, a dodawanie zostaje zastąpione operacją brania maksimum. Istnieją dwie metody kodowania. Opisana właśnie metoda to tak zwana kompozycja max-min. Jest używany do uzyskania macierzy wag połączeń, a także do uzyskania wyjść neuronów w sieci neuronowej pamięci rozmytej asocjacyjnej. Druga metoda nazywa się kodowaniem iloczynu korelacji. Uzyskuje się ją w ten sam sposób, w jaki uzyskuje się macierz wag połączeń BAM. W praktyce często stosuje się skład max-min i skoncentrujemy naszą uwagę na tej metodzie.

Przykład kodowania

Załóżmy, że dwa zbiory rozmyte, których używamy do kodowania, mają wektory dopasowania (0,3, 0,7, 0,4, 0,2) i (0,4, 0,3, 0,9). Następnie macierz W uzyskuje się stosując skład max-min w następujący sposób.

```
   0.3 [0.4 0.3 0.9] min(0.3,0.4) min(0.3,0.3) min(0.3,0.9) 0.3 0.3 0.3
W=0.7               =min(0.7,0.4) min(0.7,0.3) min(0.7,0.9)=0.4 0.3 0.7
   0.4               min(0.4,0.4) min(0.4,0.3) min(0.4,0.9) 0.4 0.3 0.4
   0.2               min(0.2,0.4) min(0.2,0.3) min(0.2,0.9) 0.2 0.2 0.2
```

Przywołaj przykład

Jeśli wprowadzimy wektor dopasowania (0,3, 0,7, 0,4, 0,2), wynik (b_1, b_2, b_3) jest określany w następujący sposób, używając bj = max(min(a_1, w_{1j}), …, min(a_m, w_{mj}) , gdzie m jest wymiarem wektora dopasowania 'a', a w_{ij} jest i-tym wierszem, j-tym elementem kolumny macierzy W.

```
b1 = max(min(0.3, 0.3), min(0.7, 0.4), min(0.4, 0.4),
     min(0.2, 0.2)) =  max(0.3, 0.4, 0.4, 0.2) = 0.4
b2 = max(min(0.3, 0.3), min(0.7, 0.3), min(0.4, 0.3),
     min(0.2, 0.2)) = max( 0.3, 0.3, 0.3, 0.2 ) = 0.3
b3 = max(min(0.3, 0.3), min(0.7, 0.7), min(0.4, 0.4),
     min(0.2, 0.2)) = max (0.3, 0.7, 0.4, 0.2) = 0.7
```

Wektor wyjściowy (0,4, 0,3, 0,7) nie jest tym samym, co drugi użyty wektor dopasowania, czyli (0,4, 0,3, 0,9), ale jest jego podzbiorem, więc przywołanie nie jest doskonałe. Jeśli wprowadzisz wektor (0,4, 0,3, 0,7) w przeciwnym kierunku, używając transpozycji macierzy W, wynik będzie (0,3, 0,7, 0,4, 0,2), pokazując rezonans. Jeśli z drugiej strony wprowadzisz (0,4, 0,3, 0,9) na tym końcu, wektor wyjściowy to (0,3, 0,7, 0,4, 0,2), co z kolei spowoduje w przeciwnym kierunku wynik (0,4, 0,3, 0,7) w tym czasie pojawia się rezonans. Czy możemy przewidzieć te wyniki? Poniższa sekcja wyjaśnia to dokładniej.

Odwołanie

Użyjmy operatora o do oznaczenia składu max - min. Doskonałe przypomnienie występuje, gdy macierz wag jest otrzymywana przy użyciu składu max-min wektorów dopasowania U i V w następujący sposób:

(i) U o W = V wtedy i tylko wtedy, gdy wzrost (U) ≥ wysokość (V).

(ii) V o WT = U wtedy i tylko wtedy, gdy wysokość (V) ≥ wysokość (U).

Zauważ również, że jeśli X i Y są dowolnymi wektorami dopasowania o takich samych wymiarach jak U i V, to:

(iii) $\mathbf{X} \circ \mathbf{W} \therefore \mathbf{V}$.
(iv) $\mathbf{Y} \circ \mathbf{W}^T \therefore \mathbf{U}$.

$A \therefore B$ to zapis mówiący, że A jest podzbiorem B.

W poprzednim przykładzie wysokość (0,3, 0,7, 0,4, 0,2) wynosi 0,7, a wysokość (0,4, 0,3, 0,9) wynosi 0,9. Zatem (0,4, 0,3, 0,9) jako nakład, wyprodukował (0,3, 0,7, 0,4, 0,2) jako wynik, ale (0,3, 0,7, 0,4, 0,2) jako nakład, dał tylko podzbiór (0,4, 0,3, 0,9). To, że zarówno (0,4, 0,3, 0,7) jak i (0,4, 0,3, 0,9) dały ten sam wynik (0,3, 0,7, 0,4, 0,2) jest zgodne z wnioskiem z powyższego, który stwierdza, że jeśli (X, Y) jest pamięcią skojarzoną rozmytą, a jeśli X jest podzbiorem X', to (X', Y) jest również pamięcią skojarzoną rozmytą.

Implementacja C++

Używamy klas, które stworzyliśmy do implementacji BAM w C++, z tym wyjątkiem, że klasę neuronu nazywamy fzneuron i nie potrzebujemy niektórych metod lub funkcji w klasie sieci. Plik nagłówkowy, plik źródłowy i dane wyjściowe z przykładowego uruchomienia programu są podane poniżej. Plik nagłówkowy nazywa się fuzzyam.hpp, a plik źródłowy nazywa się fuzzyam.cpp.

Szczegóły programu

Szczegóły programu są analogiczne do szczegółów programu podanych w rozdziale 8. Obliczenia są wykonywane za pomocą logiki rozmytej. W przeciwieństwie do wersji nierozmytej, używana jest tutaj pojedyncza przykładowa para wektorów rozmytych. Nie ma transformacji do wersji bipolarnych, ponieważ wektory są rozmyte, a nie binarne i ostre. Neuron w pierwszej warstwie to anrn, a liczba neuronów w tej warstwie to anmbr. bnrn to nazwa, którą nadajemy tablicy neuronów w drugiej warstwie, a bnmbr oznacza rozmiar tej tablicy. Kolejność operacji w programie jest następująca:

• Prosimy użytkownika o wprowadzenie przykładowej pary wektorów.

• Nadajemy sieci wektor X, w przykładowej parze.

Znajdź aktywacje elementów tablicy bnrn i otrzymaj 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, aby uzyskać aktywacje, a następnie znaleźć wektor wyjściowy, w odpowiednich kierunkach sieci pamięci asocjacyjnej rozmytej.

Duża część kodu z dwukierunkowej pamięci asocjacyjnej (BAM) jest używana przez FAM. Oto wykazy z dodanymi komentarzami, w których występują różnice między tym kodem a kodem BAM z Części 8.

Plik nagłówka

Listing 1 fuzzyam.h

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

#include <iostream.h>

#define MXSIZ 10

class fzneuron

{

protected:
```

```
int nnbr;
int inn,outn;
float output;
float activation;
float outwt[MXSIZ];
char *name;
friend class network;
public:
fzneuron() { };
void getnrn(int,int,int,char *);
};
class exemplar
{
protected:
int xdim,ydim;
float v1[MXSIZ],v2[MXSIZ]; // this is different from BAM
friend class network;
public:
exemplar() { };
void getexmplr(int,int,float *,float *);
void prexmplr();
};
class asscpair
{
protected:
int xdim,ydim,idn;
float v1[MXSIZ],v2[MXSIZ];
friend class network;
public:
asscpair() { };
void getasscpair(int,int,int);
```

```
void prasscpair();
};
class potlpair
{
protected:
int xdim,ydim;
float 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;
fzneuron (anrn)[MXSIZ],(bnrn)[MXSIZ];
exemplar (e)[MXSIZ];
asscpair (as)[MXSIZ];
potlpair (pp)[MXSIZ];
float outs1[MXSIZ],outs2[MXSIZ]; // change from BAM to floats
double mtrx1[MXSIZ][MXSIZ],mtrx2[MXSIZ][MXSIZ]; // change from
BAM to doubles
network() { };
void getnwk(int,int,int,float [][6],float [][4]);
void compr1(int,int);
void compr2(int,int);
void prwts();
void iterate();
void findassc(float *);
```

```
void asgninpt(float *);
void asgnvect(int,float *,float *);
void comput1();
void comput2();
void prstatus();
};
```

Plik źródłowy

Listing 9.2 fuzzyam.cpp

```
//fuzzyam.cpp V. Rao, H. Rao
#include "fuzzyam.h"
float max(float x,float y) //nowość w FAM
{
unosić się;
u = ((x>y) ? x : y );
wróć do ciebie;
}
float min(float x,float y) // nowość w FAM
{
unosić się;
u =( (x>y) ? y : x) ;
wróć do ciebie;
}
void fzneuron::getnrn(int m1,int m2,int m3,char *y)
{
wew;
nazwa = y;
nnbr = m1;
outn = m2;
karczma = m3;
dla(i=0;i<outn;++i){
```

```
outwt[i] = 0 ;
}
wyjście = 0;
aktywacja = 0;
}
void exemplar::getexmplr(int k,int l,float *b1,float *b2) // zmieniono
z BAM
{
int i2;
xdim = k;
ydim = l;
dla(i2=0;i2<xdim;++i2){
v1[i2] = b1[i2]; }
dla(i2=0;i2<ydim;++i2){
v2[i2] = b2[i2]; }
}
pusty przykład::prexmplr()
{
wew;
cout<<"\nX wektor, który podałeś to:\n";
dla(i=0;i<xdim;++i){
cout<<v1[i]<<" ";}
cout<<"\nY wektor, który podałeś to:\n";
dla(i=0;i<ydim;++i){
cout<<v2[i]<<" ";}
cout<<"\n";
}
void asscpair::getasscpair(int i,int j,int k)
{
idn = ja;
xdim = j;
```

```
ydim = k;
}
void asscpair::prasscpair()
{
wew;
cout<<"\nX wektor w powiązanej parze nr "<<idn<<"is:\n";
dla(i=0;i<xdim;++i){
cout<<v1[i]<<" ";}
cout<<"\nWektor Y w powiązanej parze nr "<<idn<<"is:\n";
dla(i=0;i<ydim;++i){
cout<<v2[i]<<" ";}
cout<<"\n";
}
void potlpair::getpotlpair(int k,int j)
{
xdim = k;
ydim = j;
}
void potlpair::prpotlpair()
{
wew;
cout<<"\nX wektor w możliwej skojarzonej parze to:\n";
dla(i=0;i<xdim;++i){
cout<<v1[i]<<" ";}
cout<<"\nY wektor w możliwej skojarzonej parze to:\n";
dla(i=0;i<ydim;++i){
cout<<v2[i]<<" ";}
cout<<"\n";
}
void network::getnwk(int k,int l,int k1,float b1[][6],float
b2[][4])
```

```
{
anmbr = k;
bmbr = l;
nexmplr = k1;
naspr = 0;
ninpt = 0;
int i,j,i2;
pływak tmp1,tmp2;
flaga =0;
char *y1="ANEURON", *y2="BNEURON" ;
for(i=0;i<nexmplr;++i){
e[i].getexmplr(anmbr,bnmbr,b1[i],b2[i]);
e[i].prexmplr();
cout<<"\n";
}
dla(i=0;i<anmbr;++i){
anrn[i].fzneuron::getnrn(i,bnmbr,0,y1);}
dla(i=0;i<bnmbr;++i){
bnrn[i].fzneuron::getnrn(i,0,anmbr,y2);}
dla(i=0;i<anmbr;++i){
for(j=0;j<bnmbr;++j){
tmp1 = 0,0;
for(i2=0;i2<nexmplr;++i2){
tmp2 = min(e[i2].v1[i],e[i2].v2[j]);
tmp1 = max(tmp1,tmp2);
}
mtrx1[i][j] = tmp1;
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(liczba zmiennoprzecinkowa *b)
{
int i,j;
cout<<"\n";
dla(i=0;i<anmbr;++i){
anrn[i].wyjście = b[i];
outs1[i] = b[i];
}
}
void network::compr1(int j,int k)
{
wew;
dla(i=0;i<anmbr;++i){
if(pp[j].v1[i] != pp[k].v1[i]) flaga = 1;
złamać;
}
}
void network::compr2(int j,int k)
{
wew;
dla(i=0;i<anmbr;++i){
if(pp[j].v2[i] != pp[k].v2[i]) flaga = 1;
złamać;}
}
void network::comput1() //zmieniono z BAM
{
intj;
```

```
for(j=0;j<bnmbr;++j){
int ii1;
pływak c1 =0.0,d1;
cout<<"\n";
for(ii1=0;ii1<anmbr;++ii1){
d1 = min(outs1[ii1],mtrx1[ii1][j]);
c1 = max(c1,d1);
}
bnrn[j].aktywacja = c1;
cout<<"\n neuron warstwy wyjściowej "<<j<<" aktywacja wynosi "
<<c1<<"\n";
bnrn[j].wyjście = bnrn[j].aktywacja;
outs2[j] = bnrn[j].wyjście;
cout<<"\n neuron warstwy wyjściowej "<<j<<" wyjście to "
<<bnrn[j].wyjście<<"\n";
}
}
void network::comput2() //zmieniono z BAM
{
wew;
dla(i=0;i<anmbr;++i){
int ii1;
pływak c1=0,0,d1;
dla(ii1=0;ii1<bnmbr;++ii1){
d1 = min(outs2[ii1],mtrx2[ii1][i]);
c1 = max(c1,d1);}
anrn[i].aktywacja = c1;
cout<<"\nneuro warstwy wejściowej "<<i<<"aktywacja to "
<<c1<<"\n";
anrn[i].wyjście = anrn[i].aktywacja;
outs1[i] = anrn[i].wyjście;
```

```
cout<<"\n neuron warstwy wejściowej "<<i<<"wyjście to "
<<anrn[i].wyjście<<"\n";
}
}
void network::asgnvect(int j1,float *b1,float *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(float *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 FUZZY AM\n\n";
for(j=1;j<=nasspr;++j){
as[j].prasscpair();
cout<<"\n";}
}
void main()
{
int ar = 6, br = 4, nex = 1;
float inptv[][6]={0.1,0.3,0.2,0.0,0.7,0.5,0.6,0.0,0.3,0.4,0.1,0.2};
float outv[][4]={0.4,0.2,0.1,0.0};
```

```
cout<<"\n\nTHIS PROGRAM IS FOR A FUZZY ASSOCIATIVE MEMORY NETWORK. THE

NETWORK \n";

cout<<"IS SET UP FOR ILLUSTRATION WITH "<<ar<<" INPUT NEURONS, AND "<<br;

cout<<" OUTPUT NEURONS.\n"<<nex<<" exemplars are used to encode \n";

static network famn;

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

famn.iterate();

famn.findassc(inptv[1]);

famn.prstatus();

}
```

Wyjście

Przykładowy przebieg poprzedniego programu wykorzystuje zbiory rozmyte z wektorami dopasowania (0,1, 0,3, 0,2, 0,0, 0,7, 0,5) i (0,4, 0,2, 0,1, 0,0). Jak można się spodziewać, zgodnie z wcześniejszą dyskusją, przypominanie nie jest doskonałe w odwrotnym kierunku, a rozmyta pamięć skojarzona składa się z par (0,1, 0,3, 0,2, 0,0, 0,4, 0,4) z (0,4, 0,2, 0,1, 0,0) i (0,1, 0,2, 0,2, 0, 0,2, 0,2) z (0,2, 0,2, 0,1, 0). Dane wyjściowe komputera są tak szczegółowe, że nie wymagają wyjaśnień.

```
THIS PROGRAM IS FOR A FUZZY ASSOCIATIVE MEMORY NETWORK. THE NETWORK IS

SET UP FOR ILLUSTRATION WITH SIX INPUT NEURONS, AND FOUR OUTPUT NEURONS.

1 exemplars are used to encode

X vector you gave is:

0.1 0.3 0.2 0 0.7 0.5

Y vector you gave is:

0.4 0.2 0.1 0

weights--input layer to output layer:

0.1 0.1 0.1 0

0.3 0.2 0.1 0

0.2 0.2 0.1 0

0 0 0 0

0.4 0.2 0.1 0

0.4 0.2 0.1 0

weights--output layer to input layer:

0.1 0.3 0.2 0 0.4 0.4
```

0.1 0.2 0.2 0 0.2 0.2

0.1 0.1 0.1 0 0.1 0.1

0 0 0 0 0 0

Input vector is:

0.1 0.3 0.2 0 0.7 0.5

output layer neuron 0 activation is 0.4

output layer neuron 0 output is 0.4

output layer neuron 1 activation is 0.2

output layer neuron 1 output is 0.2

output layer neuron 2 activation is 0.1

output layer neuron 2 output is 0.1

output layer neuron 3 activation is 0

output layer neuron 3 output is 0

X vector in possible associated pair is:

0.1 0.3 0.2 0 0.7 0.5

Y vector in possible associated pair is:

0.4 0.2 0.1 0

input layer neuron 0 activation is 0.1

input layer neuron 0 output is 0.1

input layer neuron 1 activation is 0.3

input layer neuron 1 output is 0.3

input layer neuron 2 activation is 0.2

input layer neuron 2 output is 0.2

input layer neuron 3 activation is 0

input layer neuron 3 output is 0

input layer neuron 4 activation is 0.4

input layer neuron 4 output is 0.4

input layer neuron 5 activation is 0.4

input layer neuron 5 output is 0.4

output layer neuron 0 activation is 0.4

output layer neuron 0 output is 0.4

output layer neuron 1 activation is 0.2

output layer neuron 1 output is 0.2

output layer neuron 2 activation is 0.1

output layer neuron 2 output is 0.1

output layer neuron 3 activation is 0

output layer neuron 3 output is 0

X vector in possible associated pair is:

0.1 0.3 0.2 0 0.4 0.4

Y vector in possible associated pair is:

0.4 0.2 0.1 0

PATTERNS ASSOCIATED:

X vector in the associated pair no. 1 is:

0.1 0.3 0.2 0 0.4 0.4

Y vector in the associated pair no. 1 is:

0.4 0.2 0.1 0

Input vector is:

0.6 0 0.3 0.4 0.1 0.2

output layer neuron 0 activation is 0.2

output layer neuron 0 output is 0.2

output layer neuron 1 activation is 0.2

output layer neuron 1 output is 0.2

output layer neuron 2 activation is 0.1

output layer neuron 2 output is 0.1

output layer neuron 3 activation is 0

output layer neuron 3 output is 0

X vector in possible associated pair is:

0.6 0 0.3 0.4 0.1 0.2

Y vector in possible associated pair is:

0.2 0.2 0.1 0

input layer neuron 0 activation is 0.1

input layer neuron 0 output is 0.1

input layer neuron 1 activation is 0.2

input layer neuron 1 output is 0.2

input layer neuron 2 activation is 0.2

input layer neuron 2 output is 0.2

input layer neuron 3 activation is 0

input layer neuron 3 output is 0

input layer neuron 4 activation is 0.2

input layer neuron 4 output is 0.2

input layer neuron 5 activation is 0.2

input layer neuron 5 output is 0.2

output layer neuron 0 activation is 0.2

output layer neuron 0 output is 0.2

output layer neuron 1 activation is 0.2

output layer neuron 1 output is 0.2

output layer neuron 2 activation is 0.1

output layer neuron 2 output is 0.1

output layer neuron 3 activation is 0

output layer neuron 3 output is 0

X vector in possible associated pair is:

0.1 0.2 0.2 0 0.2 0.2

Y vector in possible associated pair is:

0.2 0.2 0.1 0

output layer neuron 0 activation is 0.2

output layer neuron 0 output is 0.2

output layer neuron 1 activation is 0.2

output layer neuron 1 output is 0.2

output layer neuron 2 activation is 0.1

output layer neuron 2 output is 0.1

output layer neuron 3 activation is 0

output layer neuron 3 output is 0

output layer neuron 0 activation is 0.2

output layer neuron 0 output is 0.2

output layer neuron 1 activation is 0.2

output layer neuron 1 output is 0.2

output layer neuron 2 activation is 0.1

output layer neuron 2 output is 0.1

output layer neuron 3 activation is 0

output layer neuron 3 output is 0

X vector in possible associated pair is:

0.1 0.2 0.2 0 0.2 0.2

Y vector in possible associated pair is:

0.2 0.2 0.1 0

PATTERNS ASSOCIATED:

X vector in the associated pair no. 2 is:

0.1 0.2 0.2 0 0.2 0.2

Y vector in the associated pair no. 2 is:

0.2 0.2 0.1 0

THE FOLLOWING ASSOCIATED PAIRS WERE FOUND BY FUZZY AM

X vector in the associated pair no. 1 is:

0.1 0.3 0.2 0 0.4 0.4

Y vector in the associated pair no. 1 is:

0.4 0.2 0.1 0

X vector in the associated pair no. 2 is:

0.1 0.2 0.2 0 0.2 0.2

Y vector in the associated pair no. 2 is:

0.2 0.2 0.1 0

Posumowanie

W tej częsci dwukierunkowe pamięci skojarzeniowe są przedstawione dla podzbiorów rozmytych. Ich rozwój jest w dużej mierze zasługą Kosko. Dzielą one cechę rezonansu między dwiema warstwami w sieci z teorią rezonansu adaptacyjnego. Mimo że istnieją połączenia w obu kierunkach między neuronami w dwóch warstwach, 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, co z kolei prowadzi do wyjścia takiego samego jak poprzednie, rezonans zostaje osiągnięty i zostaje znaleziona powiązana para. W przypadku dwukierunkowych pamięci asocjacyjnych rozmytych jedna para zbiorów rozmytych określa jeden system pamięci rozmytej asocjacyjnej.

Wektory dopasowania są używane w kompozycji max-min. Przywołanie idealne w obu kierunkach nie jest możliwe, chyba że wysokości obu wektorów dopasowania są równe. Rozmyte pamięci asocjacyjne mogą poprawić wydajność systemu eksperckiego, zezwalając na reguły rozmyte.