Introduction to Soft Computing Assignment 2

Student: 410821305薛祖恩

Q:

Please choose Exercise 4 or 5 (p.51, Concepts of Soft Computing) to generate the fuzzy set B on f(x1, x2, …, xr) = x12 + x22 + … + xr2.

The boundary of your program function is r = 2.

Please include your codes and the results of execution in a file.

A:

In this assignment, I choose Exercise 5 on page 51. Which is “Based on the extension principle, compute the fuzzy image of given fuzzy sets using C programming.” Additionally, my code is suitable under any conditions, and the results of execution are on the last page. My submission are as follows:

# 410821305 Tsu-En Hsueh

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

void enter\_print(int i)

{

printf("\n");

if((i + 1) % 10 == 1) {

printf("--- Please enter the %dst set ---\n", (i + 1));

}

else if((i + 1) % 10 == 2) {

printf("--- Please enter the %dnd set ---\n", (i + 1));

}

else if((i + 1) % 10 == 3) {

printf("--- Please enter the %drd set ---\n", (i + 1));

}

else {

printf("--- Please enter the %dth set ---\n", (i + 1));

}

return;

}

float min(float num1, float num2)

{

return (num1 < num2) ? num1 : num2;

}

float max(float num1, float num2)

{

return (num1 > num2) ? num1 : num2;

}

int main()

{

int r;

printf("Please enter the number of sets: ");

scanf("%d", &r);

int row;

printf("Please enter the row of each set: ");

scanf("%d", &row);

// There are r elements in data

// Each element is a fuzzy set

// There are row \* 2 elements in the fuzzy set

// Dynamically allocates the r \* row \* 2 size array

float\*\*\* data = (float\*\*\*)malloc(r \* sizeof(float\*\*));

for(int i = 0; i < r; i++) {

data[i] = (float \*\*)malloc(row \* sizeof(float \*));

for(int j = 0; j < row; j++) {

data[i][j] = (float\*)malloc(2 \* sizeof(float));

}

}

// set all element to 0

for(int i = 0; i < r; i++) {

for(int j = 0; j < row; j++) {

for(int k = 0; k < 2; k++) {

data[i][j][k] = 0;

}

}

}

// user inputs

for(int i = 0; i < r; i++) {

enter\_print(i);

for(int j = 0; j < row; j++) {

printf("row: %d: ", (j + 1));

scanf("%f %f", &data[i][j][0], &data[i][j][1]);

}

}

// main part

int domain = pow(row, r);

float \*BTier = (float\*)malloc(domain \* sizeof(float));

float \*uBTier = (float\*)malloc(domain \* sizeof(float));

float \*answerBTier = (float\*)malloc(domain \* sizeof(float));

float \*answerUBTier = (float\*)malloc(domain \* sizeof(float));

int\* rowBasedIndex = (int\*)malloc(r \* sizeof(int));

// initialize BTier and uBTier

for(int i = 0; i < domain; i++) {

BTier[i] = 0;

uBTier[i] = 1.1;

answerBTier[i] = -1;

answerUBTier[i] = -1;

}

// for each iteration

for(int i = 0; i < domain; i++) {

// 10-based to row-based

for(int j = 0; j < r; j++) {

rowBasedIndex[j] = 0;

}

int decimalNum = i, index = 0;

while(decimalNum > 0) {

rowBasedIndex[index] = decimalNum % row;

index += 1;

decimalNum /= row;

}

for(int j = r - 1; j >= 0; j--) {

// according to the row-based system, calculate B~ and ,uB~

int set = r - j - 1;

int index = rowBasedIndex[j];

BTier[i] += data[set][index][0] \* data[set][index][0];

uBTier[i] = min(uBTier[i], data[set][index][1]);

}

for(int j = 0; j < domain; j++) {

if(answerBTier[j] == -1) {

answerBTier[j] = BTier[i];

answerUBTier[j] = uBTier[i];

break;

}

if(answerBTier[j] == BTier[i]) {

answerUBTier[j] = max(answerUBTier[j], uBTier[i]);

break;

}

}

}

// sorting

for(int i = 0; i < domain; i++) {

if(answerBTier[i] == -1) {

break;

}

for(int j = i + 1; j < domain; j++) {

if(answerBTier[j] == -1) {

break;

}

if(answerBTier[i] > answerBTier[j]) {

float temp = answerBTier[i];

answerBTier[i] = answerBTier[j];

answerBTier[j] = temp;

temp = answerUBTier[i];

answerUBTier[i] = answerUBTier[j];

answerUBTier[j] = temp;

}

}

}

printf("\nThe answer of the extension principle is:\nB-Tier = {\n");

for(int i = 0; i < domain, answerBTier[i] != -1; i++) {

if(answerBTier[i + 1] == -1) {

printf(" (%f, %f)\n}", answerBTier[i], answerUBTier[i]);

break;

}

printf(" (%f, %f),\n", answerBTier[i], answerUBTier[i]);

}

return 0;

}

Here are the results of the sample input and output:

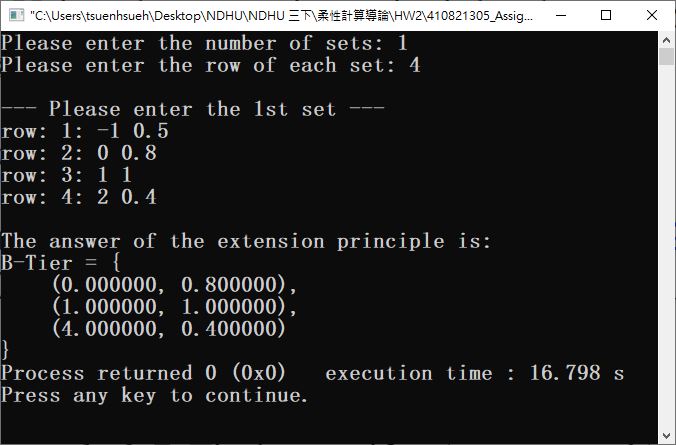


Figure 1. example 2.6 on page 49

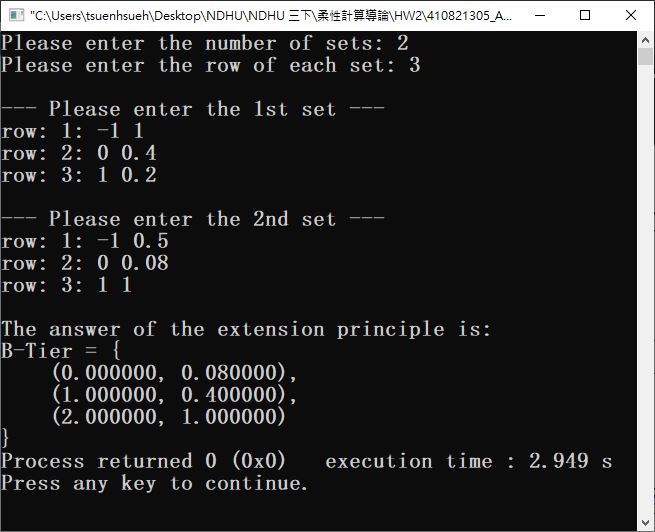


Figure 2. example 2.7 on page 49