Цель работы- построить MLP (10:20:10) по выдаче рекомендаций по покупке товара на основании корзины покупателя не используя библиотеки по NN.

**Например: Если у покупателя в корзинке гвозди и молоток, программа должна предложить гвоздодер))).**

Задачи:

1. Определить основные параметры и переменные
2. Написать блок обучения
3. Написать рабочий блок
4. Связать блок обучения и рабочий блок
5. Тестить

**Инструкция**

Задача 1:

В нашей работе на понадобятся следующие переменные и константы:

1. Вектор входных параметров
2. Вектор скрытого слоя
3. Вектор выходных параметров
4. Двумерный массив весов от входного слоя до скрытого
5. Двумерный массив весов от скрытого слоя до выходного
6. Вектор смещения скрытого слоя
7. Вектор смещения выходного слоя
8. Шаг обученя
9. Максимально допустимая ошибка
10. Сигма ноликовая
11. Вектор Сигм для скрытых слоёв
12. Счетчик максимального числа итераций обучения на одном примере

**Пример:**

float input[10];

float output[10];

float weight1[20][10] = { { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

float weight2[20][10] = { { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 } };

float hiden[20] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

float weight\_sm1[20] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

float weight\_sm2[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

float error = 0.95;

int count = 100;

float step = 0.5;

float sigma\_null

float sigma\_hiden[20] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

Задание 2:

Создаем функцию обучение **например**:(int learning()).

Локальные переменные: n- номер желаемого выхода

С,d-счетные переменные

1. Вводим обучающий пример и помещаем его в вектор input
2. Вводи номер желаемого выхода и помещаем в переменную **n**
3. Осуществляем проход от входного слоя до скрытого
4. Осуществляем проход от скрытого слоя до выходного
5. Сравниваем значение выхода под номером **n** с заданным значение ошибки **error ,** если оно меньше то переходим на следующий шаг, иначе пункт 13
6. Начинаем обратное распространение: вычисляем сигма нулевое (**sigma\_null**)
7. Вычисляем сигму для скрытых слоев
8. Корректируем значение весов от скрытого слоя до выходного
9. Корректируем значение весов от входного слоя до скрытого
10. Корректируем значение весов смещения в выходном слое
11. Корректируем значение весов смещения в скрытом слое
12. Переходим в п.3
13. Есть еще вектора обучения ? Да-П.1…..Нет-выход в главную функция (для тестов)

**Пример:**

for (int c = 0; c < 20; c++) // проход до скрытого слоя

{

for (int d = 0; d < 10; d++)

hiden[c] = hiden[c] + input[d] + weight1[c][d] + weight\_sm1[c];

hiden[c] = 1 / (1 + (exp(-hiden[c])));

}

for (int c = 0; c < 20; c++) // проход до выходного слоя

{

for (int d = 0; d < 10; d++)

output[c] = output[c] + input[d] + weight2[c][d] + weight\_sm2[c];

output[c] = 1 / (1 + (exp (-output[c])));

cout << output[c];

}

cout << output[n] << '\n';

if (output[n] < error)

{

float sigma\_null = (error - output[n]) \* output[n] \* (1 - output[n]);

float sigma\_hiden[20] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

for (int c = 0; c < 20; c++)

sigma\_hiden[c] = (sigma\_null \* weight2[c][n]) \* hiden[c] \* (1 - hiden[c]);

for (int c = 0; c < 20; c++)

weight2[c][n] = weight2[c][n] + (step \* sigma\_null \* hiden[c]);

for (int c = 0; c < 20; c++)

weight1[c][n] = weight1[c][n] + (step \* sigma\_hiden[c] \* input[c]);

for (int c = 0; c < 10; c++)

weight\_sm2[c] = weight\_sm2[c] + (step \* sigma\_null \* hiden[c]);

for (int c = 0; c < 20; c++)

weight\_sm1[c] = weight\_sm1[c] + (step \* sigma\_hiden[c] \* hiden[c]);

goto metka;

Задание 3:

1. Создаём функцию рабочая (work)
2. Вводим пример и помещаем его в вектор input
3. Осуществляем проход от входного слоя до скрытого
4. Осуществляем проход от скрытого слоя до выходного
5. Находим номер выхода с максимальным значение и выводим его на экран

Пример:

for (int c = 0; c < 20; c++) // проход до скрытого слоя

{

for (int d = 0; d < 10; d++)

hiden[c] = hiden[c] + input[d] + weight1[c][d] +

weight\_sm1[c];

hiden[c] = 1 / (1 + exp(-hiden[c]));

}

for (int c = 0; c < 20; c++) // проход до выходного слоя

{

for (int d = 0; d < 10; d++)

output[c] = output[c] + input[d] + weight2[c][d] + weight\_sm2[c];

output[c] = 1 / (1 + exp(-hiden[c]));

}

float imax = 0;

float max = output[0];

for (int c = 0; c < 20; c++)

if (output[c] > max)

{

max = output[c];

imax = c;

}

cout << imax;

}

Задание 4

В данном задании нам нужно осуществить диалоговое окно между программой и пользователем для того чтобы можно было переходить из блока обучения к рабочему блоку или к выходу из программы.

Это можно реализовать как с помощью switсh (аргумент) или для самых изащренных программистов через if.

**Пример:**

int main()

{

setlocale(LC\_ALL, "Russian");

char e;

int j;

bool k = true;

while (k)

{

cout << "Введите 1 для обучения, 2 для работы \n";

cin >> e;

switch (e)

{

case '1':

j = learning();

break;

case '2':

work();

break;

case '3':

return 0;

}

}

}

Задание 5

Придумать обучающие множество и тестовое множество . В примере красным помечен желаемый выход

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Молоток | Гвозди | Отвертка | Шурупы | Гвоздодер | Пила | Бруски | Стремянка | Краска | Кисти |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Тестовые: