**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №3.1-3.3**

з дисципліни  
«Інтелектуальні вбудовані системи»

на тему  
«Реалізація задачі розкладання числа на прості множники. Дослідження нейронних мереж. Модель perceptron. Генетичний алгоритм»

Виконала: Перевірив:

студентка групи ІП-84 Регіда П. Г.  
Д'яконенко Дар'я   
Номер заліковки: 8406

Київ 2021

**Основні теоретичні відомості 3.1**

Метод факторизації Ферма. Ідея алгоритму заключається в пошуку таких чисел А і В, щоб факторизоване число n мало вигляд: n = A 2 – B 2 . Даний метод гарний тим, що реалізується без використання операцій ділення, а лише з операціями додавання й віднімання. Приклад алгоритму: Початкова установка: x = [√𝑛] – найменше число, при якому різниця x 2 -n невід’ємна. Для кожного значення k Є ℕ, починаючи з k = 1, обчислюємо (⌈√𝑛⌉ + 𝑘) 2 − 𝑛 і перевіряємо чи не є це число точним квадратом. • Якщо не є, то k++ і переходимо на наступну ітерацію. • Якщо є точним квадратом, тобто 𝑥 2 − 𝑛 = (⌈√𝑛⌉ + 𝑘) 2 − 𝑛 = 𝑦 2 , то ми отримуємо розкладання: 𝑛 = 𝑥 2 − 𝑦 2 = (𝑥 + 𝑦)(𝑥 − 𝑦) = 𝐴 ∗ 𝐵, в яких 𝑥 = (⌈√𝑛⌉ + 𝑘) Якщо воно є тривіальним і єдиним, то n – просте.

**Лістинг програми 3.1**

static ulong[] FermatFactor(ulong n)

{

ulong a, b;

if ((n % 2UL) == 0)

{

return new[] { 2UL, n / 2UL };

}

a = Convert.ToUInt64(Math.Ceiling(Math.Sqrt(n)));

if (a \* a == n)

{

return new[] { a, a };

}

while (true)

{

ulong tmp = a \* a - n;

b = Convert.ToUInt64(Math.Sqrt(tmp));

if (b \* b == tmp)

{

break;

}

a++;

}

return new[] { a - b, a + b };

}

**Результат виконання програми**



**Основні теоретичні відомості 3.2**

Важливою задачеюяку система реального часу має вирішувати є отримання необхідних для обчислень параметрів, її обробка та виведення результату у встановлений дедлайн. З цього постає проблема отримання водночас точних та швидких результатів. Модель Перцпептрон дозволяє покроково наближати початкові значення.

**Лістинг програми 3.2**

private void percButton\_Click(object sender, EventArgs e)

{

double[,] data = new double[4, 2]

{ { 0, 6 },

{ 1, 5 },

{ 3, 3 },

{ 2, 4 }

};

float speed = float.Parse(percSigma.Text);

int iterations = int.Parse(percIterations.Text);

float p = 4;

float y, delta;

float w1 = 0;

float w2 = 0;

int i = 0;

while (i < iterations)

{

Console.WriteLine(i % 4);

double[] curPoint = { data[i % 4, 0], data[i % 4, 1] };

y = Convert.ToSingle(curPoint[0]) \* w1 + Convert.ToSingle(curPoint[1]) \* w2;

if ((y > p && (i % 4 == 0 || i % 4 == 1)) || (y < p && (i % 4 == 2 || i % 4 == 3)))

{

bool res = true;

for(int j = 0; j<4; j++)

{

if (j < 2)

{

res = (Convert.ToSingle(data[j, 0]) \* w1 + Convert.ToSingle(data[j, 1]) \* w2) > p;

} else

{

res = (Convert.ToSingle(data[j, 0]) \* w1 + Convert.ToSingle(data[j, 1]) \* w2) < p;

}

if (!res)

{

break;

}

}

if (res)

{

percResult.Text = string.Concat(w1.ToString(), ", ", w2.ToString());

}

}

delta = p - y;

w1 = w1 + delta \* speed \* Convert.ToSingle(curPoint[0]);

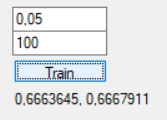
w2 = w2 + delta \* speed \* Convert.ToSingle(curPoint[1]);

i += 1;

}

}

**Результат виконання програми**



**Основні теоретичні відомості 3.3**

Генетичні алгоритми служать, головним чином, для пошуку рішень в багатовимірних просторах пошуку.

Можна виділити наступні етапи генетичного алгоритму: • (Початок циклу)

• Розмноження (схрещування)

• Мутація

• Обчислити значення цільової функції для всіх особин

• Формування нового покоління (селекція)

• Якщо виконуються умови зупинки, то (кінець циклу), інакше (початок циклу).

**Лістинг програми 3.3**

string res;

int a = int.Parse(geneticA.Text);

int b = int.Parse(geneticB.Text);

int c = int.Parse(geneticC.Text);

int d = int.Parse(geneticD.Text);

int y = int.Parse(geneticY.Text);

long start = DateTime.Now.Millisecond;

for (int i = 0; i < population; i++)

{

gens[i] = new int[4];

gens[i][0] = (new Random().Next(1, max));

gens[i][1] = (new Random().Next(1, max));

gens[i][2] = (new Random().Next(1, max));

gens[i][3] = (new Random().Next(1, max));

}

int index = -1;

for (int i = 0; i < population; ++i)

{

int result = a \* gens[i][0] + b \* gens[i][1] + c \* gens[i][2] + d \* gens[i][3];

deltas[i] = Math.Abs(result - y);

if (deltas[i] == 0) index = i;

}

if (index != -1)

{

geneticRes.Text = "x1 = " + gens[index][0].ToString() + " x2 = " + gens[index][1] + " x3 = " + gens[index][2] + " x4 = " + gens[index][3] + " delta = " + deltas[index];

}

long end = DateTime.Now.Millisecond;

while (end - start < time)

{

Console.WriteLine(end-start + " " + time);

double allSurvival = 0;

for (int i = 0; i < population; i++)

{

allSurvival += 1 / deltas[i];

}

for (int i = 0; i < population; i++)

{

survival[i] = (1 / deltas[i]) / allSurvival;

}

int father = getParent();

int mother = getParent();

int[][] children = new int[50][];

for(int i = 0; i < population; i++)

{

children[i] = new int[4];

children[i][0] = gens[father][0];

children[i][1] = gens[father][1];

children[i][2] = gens[father][2];

children[i][3] = gens[father][3];

}

gens = children;

index = -1;

for (int i = 0; i < population; ++i)

{

int result = a \* gens[i][0] + b \* gens[i][1] + c \* gens[i][2] + d \* gens[i][3];

deltas[i] = Math.Abs(result - y);

if (deltas[i] == 0) index = i;

}

if (index != -1)

{

geneticRes.Text = "x1 = " + children[index][0] + " x2 = " + children[index][1] + " x3 = " + children[index][2] + " x4 = " + children[index][3] + " delta = " + deltas[index];

}

end = DateTime.Now.Millisecond;

}

var smallDelta = Double.PositiveInfinity;

int deltaIndex = 0;

for(int i = 0; i < population; i++)

{

if (deltas[i] < smallDelta)

{

smallDelta = deltas[i];

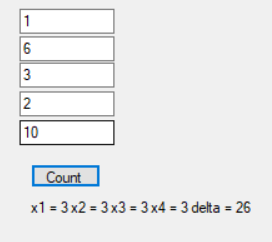
deltaIndex = i;

}

}

geneticRes.Text = "x1 = " + gens[deltaIndex][0].ToString() + " x2 = " + gens[deltaIndex][1] + " x3 = " + gens[deltaIndex][2] + " x4 = " + gens[deltaIndex][3] + " delta = " + deltas[deltaIndex];

**Результат виконання програми**



**Висновки**

Під час виконання лабораторної роботи я ознайомилася з основними принципами розкладання числа на прості множники та принципами машинного навчання за допомогою математичної моделі Перцептрон, вирішила діафантове рівняння за допомогою генетичного алгоритмую