**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 №3.1/3.2**

з дисципліни  
«Інтелектуальні вбудовані системи»

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

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 3.1**

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]);

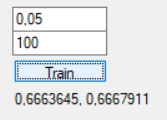
w2 = w2 + delta \* speed \* Convert.ToSingle(curPoint[1]);

i += 1;

}

}

**Результат виконання програми**



**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.