**Лабораторна робота** **№ 3**

**Тема**: навчання моделей за опорними функціями Колмогорова–Габора.

**Мета**: закріпити навички використання алгоритмів навчання моделей за опорними функціями Колмогорова–Габора в індуктивних алгоритмах обробки інформації.

**Хід роботи**

Відпрацьовується алгоритм навчання моделей за опорними функціями Колмогорова–Габора та його використання в алгоритмах індуктивного моделювання.

**Завдання до лабораторної роботи**

Спроектувати та програмно реалізувати модуль навчання моделей за опорними функціями Колмогорова–Габора. Протестувати створену систему за індивідуальними даними.

Процес розробки прогнозуючої моделі складається із таких етапів:

1. Визначення вигляду моделі.

2. Створення первинного опису функції за відомими її значеннями.

3. Складення системи умовних рівнянь Гауса.

4. Нормалізація системи умовних рівнянь. В результаті отримуємо систему

нормальних рівнянь Гауса.

5. Визначення невідомих коефіцієнтів прогнозуючої моделі, тобто визначення

моделі.

6. Визначення періоду прогнозування отриманої моделі.

Візьмемо дані про ціну китайского юаня відносно рубля за останній тиждень.

|  |  |
| --- | --- |
| Номер | Ціна |
| 1 | 11.36 |
| 2 | 11.4 |
| 3 | 11.4 |
| 4 | 11.41 |
| 5 | 11.41 |
| 6 | 11.42 |
| 7 | 11.44 |

1. Функція яка визначає складає систему умовних рівнянь Гауса:

private static Double[][] polynomial(List<Double> values) {

Double[][] F = new Double[values.size() - 2][5];

for (inti = 0; i<F.length; i++) {

F[i][0] = values.get(i + 2);

F[i][1] = 1.;

F[i][2] = values.get(i + 1);

F[i][3] = values.get(i);

F[i][4] = values.get(i) \* values.get(i + 1);

}

return F;

}

1. Функція яка нормалізує систему умовних рівнянь:

private static Double[][] normalization(Double[][] F, int m) {

doubleaj;

int row = 0;

intiter = 1;

Double[][] normMass = new Double[m - 1][5];

Double[][] middleMas = new Double[m][5];

for (inti = 0; i<normMass.length; i++) {

for (int j = 0; j < 5; j++) {

normMass[i][j] = 0.;

}

}

for (int k = 0; k < m - 1; k++) {

for (inti = 0; i<F.length; i++) {

aj = F[i][iter];

for (int j = 0; j < F[i].length; j++) {

middleMas[i][j] = F[i][j] \* aj;

}

}

System.out.println();

for (inti = 0; i<middleMas.length; i++) {

for (int j = 0; j < 5; j++) {

System.out.print(middleMas[i][j] + " ");

}

System.out.println();

}

iter++;

List<Double> elements = getElements(middleMas);

for (int j = 0; j < 5; j++) {

normMass[row][j] = elements.get(j);

}

if (row < m) {

row++;

}

}

System.out.println();

for (inti = 0; i<normMass.length; i++) {

for (int j = 0; j < 5; j++) {

System.out.print(normMass[i][j] + " ");

}

System.out.println();

}

returnnormMass;

}

1. Функція яка сумує усі перемножені рівняння:

private static List<Double>getElements(Double[][] mass) {

List<Double> elements = new ArrayList<>();

for (inti = 0; i<mass.length; i++) {

double flag = 0;

for (int j = 0; j < 5; j++) {

flag += mass[j][i];

}

elements.add(flag);

}

return elements;

}

1. Клас Applicationв якому відбувається покрокове змінення заданих значень функції за алгоритмом процесу розробки прогнозуючої моделі:

Данні які введено в колекцію valuesOfFuncвзяті з практичної роботи №5.

public class Application {

public static void main(String[] args) {

List<Double>valuesOfFunc = new ArrayList<>();

//Заповнюємо масив значеннями ціни за кожний день :

valuesOfFunc.add(11.36);

valuesOfFunc.add(11.4);

valuesOfFunc.add(11.4);

valuesOfFunc.add(11.41);

valuesOfFunc.add(11.41);

valuesOfFunc.add(11.42);

valuesOfFunc.add(11.44);

int rows = valuesOfFunc.size() - 2;

//Заповнюємо масив умовними рівняннями Гауса :

Double[][] F = polynomial(valuesOfFunc);

for (Double[] aF : F) {

for (Double anAF : aF) {

System.out.print(anAF + " ");

}

System.out.println();

}

//Нормалізуємо систему умовних рівнянь :

Double[][] normF = normalization(F, rows);

double[] B = new double[rows - 1];

for (inti = 0; i< rows - 1; i++) {

for (int j = 0; j < 1; j++) {

B[i] = normF[i][j];

}

}

double[][] A = new double[rows - 1][rows - 1];

for (inti = 0; i<A.length; i++) {

for (int j = 0; j < A[i].length; j++) {

A[i][j] = normF[i][j+1];

}

}

//Знаходимо невідомі коефіцієнти прогнозуючої моделі

RealMatrix coefficients = MatrixUtils.createRealMatrix(A);

DecompositionSolver solver = new LUDecomposition(coefficients).getSolver();

RealVector constants = MatrixUtils.createRealVector(B);

RealVector solution = solver.solve(constants);

double[] X = new double[4];

for(inti = 0; i<X.length; i++) {

X[i] = solution.getEntry(i);

}

for (inti = 0; i<X.length; i++) {

System.out.println("X[" + (i+1) + "] = " + X[i]) ;

}

//Знаходимо F – прогнозуюче :

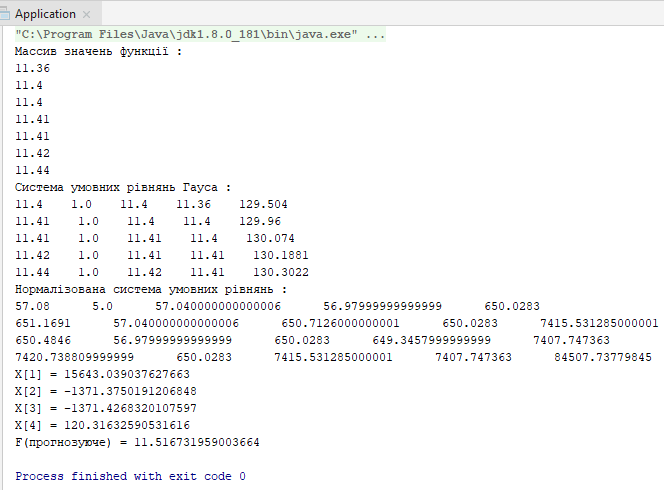
double Fi = X[0] + X[1] \* valuesOfFunc.get(valuesOfFunc.size() - 1) + X[2] \* valuesOfFunc.get(valuesOfFunc.size() - 2) + X[3] \* valuesOfFunc.get(valuesOfFunc.size() - 1) \* valuesOfFunc.get(valuesOfFunc.size() - 2);

System.out.println("Fi = " + Fi);

}

}

Вихідні данні програми:



**Висновок.** На цій лабораторній роботі ми визначили F – прогнозоване, яке відрізняється від F–реального з похибкою , за допомогою опорних функцій Колмогорова-Габора.