```
/*
 * Written by Masaki Uto (2017/04/1)
 * The following libraries are required.
 * 1. commons-math3-3.2.jar
 */
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.Arrays;

import org.apache.commons.math3.linear.MatrixUtils;
import org.apache.commons.math3.linear.RealMatrix;
public class MaximumLikelihood { // サンプルサイズの指定
        final double cond = 0.001;
        // 学習率
        final double eta = 1.0;
        private double[] X;
        private double[] Y;
        //a, bの推定値
        private double[] eW;
        //サンプルサイズ
        int N = 5000;

        MaximumLikelihood() throws IOException {
                X = new double[N];
                Y = new double[N];
                init();
                runNewton();
        }

        void init() throws IOException {
                readData("data.csv");
                eW = new double[2];
        }

        void readData(String filename) throws IOException {
                BufferedReader br = new BufferedReader(new
FileReader(new File(filename)));
                br.readLine();
                int i = 0;
                while(true){
                        String line = br.readLine();
                        if(line == null) break;
                        String[] data = line.split(",").clone();
                        X[i] = Double.parseDouble(data[0]);
                        Y[i] = Double.parseDouble(data[1]);
                        ++i;
                }
                br.close();
        }

        double[] getGradient() {
```

```
                double[] grad = new double[2];
                /*
                 *
                 * 勾配を求めてgradに入力.
                 *
                 *
                 */
                return grad;
        }

        double[][] getInvHesse() {
                double[][] Hesse = new double[2][2];
                /*
                 *
                 *
                 * ヘッセ行列を求めてHesseに入力.
                 *
                 */

                double[][] invHesse = new double[4][4];
                RealMatrix mA =
MatrixUtils.createRealMatrix(Hesse); RealMatrix m =
MatrixUtils.blockInverse(mA,0);
                for (int i = 0; i < m.getRowDimension(); i++) {
                        for (int j = 0; j <
m.getColumnDimension(); j++) {
                                invHesse[i][j] = m.getEntry(i, j);
                        }
                }
                return invHesse;
        }

        void runNewton() {
                while (true) {
                        double[] g = getGradient();
                        double[][] invH = getInvHesse();
                        boolean flag = true;
                        double[] delta = new double[eW.length];//
deltaはInvHとgの積
                        /*
                         *
                         * deltaを求めてeWを更新
                         *
                         */
                        printCurrentParam();
                        if (flag) break;
                }
        }

        void printCurrentParam() {

System.out.println(Arrays.toString(eW).replace("[","").replace("]","
"));
```

```
        }

        public static void main(String[] args) throws IOException {
                new MaximumLikelihood();
        }
}
```