```
package mcmc;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;

import org.uncommons.maths.random.GaussianGenerator;

public class mcmc {
    final int n = 5000;
    final int nParam = 4;
    final String filename = "data.csv";
    private Random rand;
    // 学習データ
    private double[][] x;
    private int[] y;
    // 推定過程における現在のパラメータ値
    private double[] param;
    // パラメータサンプル
    private ArrayList<double[]> params;
    // MCMCの各種設定
    final int MaxLoop = 2000;
    final int burnin = 1000;
    final int interval = 100;
    final double sigma = 0.01;
    // ハイパーパラメータ
    final double[] prior = {0.0, 1.0};

    mcmc() throws IOException{
        init();
        FileReader(filename);
        runMCMC();
        double[] eap = getEap();
        printResult(eap);
    }

    // 各変数の初期化
    void init(){
        rand = new Random();
        x = new double[n][nParam-1];
        y = new int[n];
        param = new double[nParam];
        params = new ArrayList<double[]>();
    }

    // データファイルの読み込み
    void FileReader(String fileName) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(new
  File(fileName)));
        br.readLine();
        for(int l=0;l<n;l++){
            String line = br.readLine();
            if(line == null) break;
            String[] d = line.split(",");
            for(int i=0;i<nParam-1;i++){
                x[l][i] = Double.valueOf(d[i]);
            }
            y[l] = Integer.valueOf(d[nParam-1]);
        }
        br.close();
    }

    void runMCMC(){
        for(int i=0;i<MaxLoop;i++){
            if(i % 500 == 0) System.out.println("Loop " + i);
            // 各パラメータを更新
            for(int j=0;j<nParam;j++){
                update(j);
            }
            if(i > burnin){
                if(i% interval == 0){
                    params.add(param.clone());
                }
            }
        }
    }

    void update(int i) {
        double _p = param[i];
        // 更新前の事後確率
        double lb = getLogLikelihood() + getLogGaussian(prior[0], prior[1],
param[i]);
        // 候補値のサンプリング
        GaussianGenerator proposal = new GaussianGenerator(param[i], sigma,
rand);
        param[i] = proposal.nextValue();
        // 更新後の事後確率
        double la = getLogLikelihood() + getLogGaussian(prior[0], prior[1],
param[i]);
        // 採択確率の計算
        double alfa = Math.exp(la - lb);
        // 採択棄却
        double t = rand.nextDouble();
        if (t > alfa) {
            param[i] = _p;
        }
    }

    // 対数尤度の計算
    double getLogLikelihood(){
        double LL = 0;
        for(int i=0;i<n;i++){
            double p = 1.0 / (1.0 + Math.exp(-(param[0]*x[i][0] + param[1]*x[i]
[1] + param[2]*x[i][2]+ param[3])));
            if(y[i] == 1) LL += Math.log(p);
            else LL += Math.log(1.0 - p);
        }
        return LL;
    }
    // 正規分布に基づく確率値の対数を取得
    double getLogGaussian(double m, double s, double v) {
        double ee = - Math.pow(v - m, 2) / (2 * s * s);
        return Math.log(Math.exp(ee) / (Math.sqrt(2 * Math.PI) * s));
    }

    // EAP推定値を計算
    double[] getEap(){
        double[] eap = new double[nParam];
        int L = params.size();
        for(int i=0;i<L;i++){
            double[] cParam = params.get(i);
            for(int j=0;j<nParam;j++){
                eap[j] += cParam[j] / L;
            }
        }
        return eap;
    }

    void printResult(double[] eap){
        System.out.println("EAP推定値:");
        System.out.println(" a = " + eap[0]);
        System.out.println(" b = " + eap[1]);
        System.out.println(" c = " + eap[2]);
        System.out.println(" d = " + eap[3]);
    }

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