Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра Программных систем и баз данных

Лабораторная работа №5

по курсу «Статистические методы анализа данных»

Факультет: ПМИ

Группа: ПМ -21, ПМ-23

Студенты: Заборцев В.А, Яковлева М.А.

Преподаватели: Попов А.А., Гультяева Т.А.

Вариант:3

Новосибирск

2015

**Задание:**

1. Сгенерировать экспериментальные данные, в которых в явном виде присутствует эффект мультиколлинеарности. Регрессия на 8 факторах. Эффект мультиколлинеарности создают 4 фактора. Имеется разброс в масштабах факторов.
2. Рассчитать ряд показателей, характеризующих эффект мультиколлинеарности, определить факторы, ответственные за неё.
3. Построить ридж-оценки параметров при различном значении параметра регуляризации, выбрать оптимальное значение. Построить графики изменения квадрата эвклидовой норы оценок параметров и остаточной суммы квадратов от параметра регуляризации.
4. Провести оценивание модели регрессии по методу главных компонентов. Перейти к описанию в исходном пространстве факторов. Сравнить решение с ридж-оцениванием по смешению оценок и точности предсказания отклика.

**Ход работы:**

1. **Генерация экспериментальных данных, с эффектом мультиколлинеарности.**

θ = (1, 1, 1, 1, 1, 1, 1, 1, 1)

f(x)=(1, x1, x2, x3, x4, x5, x6, x7, x1 + 2x2 - 3x3 + )

Число экспериментов n=50

1. **Расчет показателей, характеризующих эффект мультиколлинеарности**

*а) Определитель информационной матрицы* :

208111,657406803

*Определитель информационной матрицы, нормированной на след:*

2,10594788500701E-25

*б) Минимальное собственное число матрицы*:

3,6371927789086E-05

Минимальное собственное значение близко к нулю, что говорит о плохой обусловленности матрицы. Чем меньше минимальное собственное значение , тем сильнее мультиколлинеарность.

*в) Мера обусловленности матрицы по Нейману-Голдстейну.*

55695589,8443597

*г) Максимальная парная сопряженность:*

Матрица сопряженности имеет вид:

, где 

Показатель мультиколлинеарности: 0,753977805095502

*д) В качестве меры мультиколлинеарности* можно взять , где  вычисляется по формуле:

, элемент матрицы, обратной к сопряженной R.

0,999998926110047

Вывод: Эффект мультиколлинеарности присутствует.

1. **Получим ридж-оценки неизвестных параметров модели при различных значениях параметров регуляризации:**

Часто матрицу Λ задают диагональной в виде: .

Будем рассматривать . Параметр выбирался исходя из стабилизации .

|  |  |  |
| --- | --- | --- |
|  | 25 | 0,000582694516423023 |
| θ | 0,0488576978343727 | 1,0169111824831 |
| 0,136217319020234 | 1,24988963656966 |
| 0,0098738732552551 | 1,77037738255404 |
| 0,00493963521618233 | 0,0924850042361898 |
| 0,0128452790028637 | 0,770872076470483 |
| 0,10528195781574 | 0,788876836455941 |
| 0,045033470211261 | 1,09048287523291 |
| 0,0511367421891399 | 0,964001020100991 |
| 0,0315644147058688 | 0,657076246657633 |

1. **Метод главных компонент.**

Перейдем к центрированным переменным:  Выразим главные компоненты в матричном виде через -матрица из всех собственных векторов матрицы . Исключим из матриц  и столбцы, соответствующие собственным значениям с незначительными вкладами . Обозначим полученные матрицы как и соответственно. И посчитаем вектор , с помощью которого найдем .

Вектор собственных значений:

|  |
| --- |
| 8,81628136944998E-06 |
| 0,027353757865203 |
| 0,0708358362045977 |
| 0,557323826306047 |
| 1,25491379989581 |
| 3,57628885926304 |
| 16,1742751212385 |
| 10,0000000848293 |
| 33,4514033173321 |

Незначительный вклад даёт , эту строчку и будем исключать.

Новый вектор оценок:

|  |
| --- |
| 0.9804785 |
| 0.5931772 |
| 1.1798288 |
| 0.9949822 |
| 0.9488221 |
| 0.9927367 |
| 1.0251115 |
| 0.7881154 |
| 1.0568525 |

RSS = 1.1117116386421335 Norm = 8.36988906606933

**Код программы:**

class Lab5 {

int countExperiments = 50;

double noize = 0.05;

double error = 0.01;

private List<double> e = new List<double>();

private List<double> u = new List<double>();

private List<double> y = new List<double>();

public List<double> RSSs = new List<double>();

public List<double> Norms = new List<double>();

public List<double> Lamdas = new List<double>();

public List<Matrix> Tettas = new List<Matrix>();

public List<Matrix> Bettas = new List<Matrix>();

public List<double> BettasScalars = new List<double>();

protected double[] Tetta = new double[] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };

protected double[] calcF(double[] x) {

return new double[9] { 1.0, x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7] };

}

private double normVector(Matrix vec) {

return (vec.Transpose() \* vec)[0, 0];

}

private double rss(Matrix x, Matrix y, Matrix tetta) {

return normVector(y - x \* tetta);

}

protected double calcU(double[] f, double[] tetta) {

double result = 0.0;

for (int i = 0; i < tetta.Count(); i++)

result += tetta[i] \* f[i];

return result;

}

private Matrix MNK(Matrix A, Matrix B, Matrix c) {

Matrix AT = A.Transpose();

return (AT \* A + B).Inverse() \* AT \* c;

}

public String GenerateData() {

String text = "";

int n = countExperiments;

int m = Tetta.Count();

Matrix X = new Matrix(n, m);

double x1 = 1.0, x2 = 0.5, x3 = 0.7, x4 = 0.4, x5 = 0.3, x6 = 1.0, x7 = 2.0;

text += "1. Generating data"+ Environment.NewLine;

double avgU = 0.0;

List<double[]> points = new List<double[]>();

for (int i = 0; i < n; i++) {

double e\_ = math.Distribution.Normal(0.0, error);

double x1\_ = math.Distribution.Flat(-x1, x1 \* 2.0);

double x2\_ = math.Distribution.Flat(-x2, x2 \* 2.0);

double x3\_ = math.Distribution.Flat(-x3, x3 \* 2.0);

double x4\_ = math.Distribution.Flat(-x4, x4 \* 2.0);

double x5\_ = math.Distribution.Flat(-x5, x5 \* 2.0);

double x6\_ = math.Distribution.Flat(-x6, x6 \* 2.0);

double x7\_ = math.Distribution.Flat(-x7, x7 \* 2.0);

double x8\_ = x1\_ + 2.0 \* x2\_ - 3.0 \* x3\_ + e\_;

//double x8\_ = math.Distribution.Flat(-x6, x6 \* 2.0);

double[] point = new double[]{

x1\_,x2\_,x3\_,x4\_,x5\_,x6\_,x7\_, x8\_ };

points.Add(point);

double[] f = calcF(point);

X[i] = new Matrix(m, f);

u.Add(calcU(f, Tetta));

avgU += u.Last();

}

avgU /= countExperiments;

double sigma2 = avgU \* noize;

for (int i = 0; i < countExperiments; i++) {

e.Add(math.Distribution.Normal(0.0, sigma2));

y.Add(u[i] + e.Last());

}

text += Environment.NewLine + "2. Calculating multi-conjugation effect data" + Environment.NewLine;

Matrix XT = X.Transpose();

Matrix XTX = XT \* X;

double detXTX = XTX.Determinant();

text += "det(XTX) = \t" + detXTX.ToString() + Environment.NewLine;

Matrix XTX\_Trace = XTX / XTX.Trace();

double detXTX\_Trace = XTX\_Trace.Determinant();

text += "det(XTX/trace) = \t" + detXTX\_Trace.ToString() + Environment.NewLine;

double maxL = XTX.MaxEiganValue();

double minL = XTX.MinEiganValue();

text += "minLambda = \t\t" + minL.ToString() + Environment.NewLine;

double lmax\_lmin = maxL / minL;

text += "Neiman-Goldstein = \t" + lmax\_lmin.ToString() + Environment.NewLine;

Matrix R = new Matrix(Tetta.Count(), Tetta.Count());

double r\_max = 0.0;

for (int j = 0; j < Tetta.Count(); j++) {

for (int i = 0; i < Tetta.Count(); i++) {

double sum\_up = 0.0, sum\_d1 = 0.0, sum\_d2 = 0.0;

for (int k = 0; k < countExperiments; k++) {

sum\_up += X[k, i] \* X[k, j];

sum\_d1 += X[k, i] \* X[k, i];

sum\_d2 += X[k, j] \* X[k, j];

}

double rij = sum\_up / (Math.Sqrt(sum\_d1) \* Math.Sqrt(sum\_d2));

if (i != j && Math.Abs(rij) > r\_max)

r\_max = Math.Abs(rij);

R[i, j] = rij;

}

R[j, j] = 1.0;

}

text += "Max pair conjugation = \t" + r\_max.ToString() + Environment.NewLine;

Matrix RInverse = R.Inverse();

r\_max = 0.0;

for (int i = 0; i < Tetta.Count(); i++) {

double ri = Math.Sqrt(1.0 - 1.0 / RInverse[i,i]);

if (ri > r\_max)

r\_max = ri;

}

text += "Max conjugation = \t" + r\_max.ToString() + Environment.NewLine;

text += Environment.NewLine + "3. Ridge assessment" + Environment.NewLine;

Matrix Z = X \* Matrix.DiagonalSqrt(XTX).Inverse();

Matrix Y = new Matrix(1, y.ToArray());

for (double lambda = 0.0; lambda <= 50; lambda += 1) {

Matrix Л = Matrix.Diagonal(XTX) \* lambda;

Tettas.Add(MNK(X, Л,Y));

RSSs.Add(rss(X, Y, Tettas.Last()));

Norms.Add(normVector(Tettas.Last()));

Lamdas.Add(lambda);

Bettas.Add(MNK(Z,lambda\*Matrix.Identity(Z.LenghtY()),Y));

BettasScalars.Add(Math.Sqrt(normVector(Bettas.Last())));

}

int center = Lamdas.Count/2;

text += "Lambda = " + Lamdas[center].ToString() + Environment.NewLine + "Tetta = " + Environment.NewLine;

for (int i = 0; i < Tettas.First().LenghtX(); i++)

text += Tettas[center][i,0].ToString() + Environment.NewLine;

Matrix allBettas = new Matrix(BettasScalars.Count, BettasScalars.ToArray());

double sigmaEst = rss(X, Y, Tettas.First()) / (n - m);

Matrix D = Matrix.DiagonalSqrt(XTX) \* Tettas.First();

double lambdaNew = m \* sigmaEst / Math.Sqrt(normVector(D));

Tettas.Add(MNK(X,Matrix.Diagonal(XTX)\*lambdaNew,Y));

RSSs.Add(rss(X, Y, Tettas.Last()));

Norms.Add(normVector(Tettas.Last()));

Lamdas.Add(lambdaNew);

text += "Lambda\* = " + lambdaNew.ToString() + Environment.NewLine + "Tetta = " + Environment.NewLine;

for (int i = 0; i < Tettas.Last().LenghtX(); i++)

text += Tettas.Last()[i, 0].ToString() + Environment.NewLine;

text += Environment.NewLine + "3. Main components" + Environment.NewLine;

double[] x\_center = new double[points[0].Count()];

double y\_center = 0.0;

for(int i = 0; i < points.Count; i++){

y\_center += y[i];

for(int j=0;j<points[i].Count();j++)

x\_center[j] += points[i][j];

}

y\_center /= (double)points.Count;

for(int i = 0; i < points[0].Count(); i++)

x\_center[i] /= (double)points.Count;

Matrix X\_center = new Matrix(n, m);

List<double> y\_cent = new List<double>();

for(int i = 0; i < points.Count; i++){

double[] point = new double[points[0].Count()];

for (int j = 0; j < points[i].Count(); j++)

point[j] = points[i][j] - x\_center[j];//X\_center[i,j] = points[i][j] - x\_center[j];

double[] f = calcF(point);

X\_center[i] = new Matrix(m, f);

y\_cent.Add(y[i] - y\_center);

}

Matrix Y\_center = new Matrix(y\_cent.Count, y\_cent.ToArray());

Matrix X\_centerT = X\_center.Transpose();

Matrix X\_centerTX\_center = X\_centerT\*X\_center;

var eiganVals = X\_centerTX\_center.EiganValues();

text += "eiganValues:" + Environment.NewLine;

Matrix nulMat = new Matrix(m,1);

List<Matrix> eiVecs = new List<Matrix>();

for (int i = eiganVals.Count() -1; i >=0; i--) {

text += eiganVals[i].ToString() + Environment.NewLine;

Matrix a\_lI = X\_centerTX\_center - eiganVals[i] \* Matrix.Identity(X\_centerTX\_center.LenghtX());

eiVecs.Add(Matrix.GaussSolve(a\_lI, nulMat));

}

Matrix V = new Matrix(m, eiganVals.Count());

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

for (int j = 0; j < eiganVals.Count(); j++)

V[i, j] = eiVecs[i][j,0];

Matrix Z\_ = X\_center \* V;

int notCool = 3;

Matrix Vr = new Matrix(m,eiganVals.Count() - notCool);

Matrix Zr = new Matrix(Z\_.LenghtX(), Z\_.LenghtY()- notCool);

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

for (int j = 0; j < eiganVals.Count() - notCool; j++)

Vr[i, j] = V[i, j];

for (int i = 0; i < Z\_.LenghtX(); i++)

for (int j = 0; j < Z\_.LenghtY() - notCool; j++)

Zr[i, j] = Z\_[i, j];

Matrix bb = (Zr.Transpose() \* Zr).Inverse() \* Zr.Transpose() \* Y\_center;

Matrix tettaCenter = Vr \* bb;

return text;

}

}