Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра теоретической и прикладной информатики

Курсовая работа

по дисциплине

«Планирование и анализ эксперимента»

Робастные методы оценивания параметров регрессии

|  |  |
| --- | --- |
| Факультет: | прикладной математики и информатики |
| Группа: | ПМ-41 |
| Студент: | Пронькин В.Ю. |
| Преподаватель: | Лисицин Д.В. |
| Вариант: | 3.2 |

Новосибирск

2018

1. **Постановка задачи**

1. Выбрать полиномиальную невысокого порядка (квадратичную, кубическую) модель зависимости отклика *y* от одного фактора *x*. Дисперсию выбрать соответствующей 5…10% мощности сигнала. По данной модели сгенерировать экспериментальные данные, содержащие 5…30% выбросов. Выбросы смоделировать, добавив к значению отклика фиксированное смещение величиной 1…4 стандартных отклонений. Проконтролировать наличие выбросов в выборке визуально.

2. Реализовать алгоритм поиска М-оценок параметров итерационным МНК для функции потерь, указанной в варианте.

3. Выбрать несколько значений параметра функции потерь (некоторые рекомендуемые значения указаны в варианте) и найти значения М-оценок для каждого из них. В качестве начального приближения оценок в алгоритме итерационного МНК можно взять истинные значения параметров или можно произвести поиск глобального оптимума путем многократного применения итерационного МНК из различных начальных приближений. Вычислить МНК-оценку. Сравнить качество всех полученных оценок по величине , выбрать значение параметра функции потерь, дающее наилучшее качество.

**Вариант 3.2:**

Оценки Шурыгина параметров регрессионной модели.

|  |  |  |
| --- | --- | --- |
| ρ(*z*) | *w*(z) | Параметр |
| , |  | 100  10  1  0.5  0  -0.5 |

1. **Полученный набор данных и значения ошибок наблюдений**

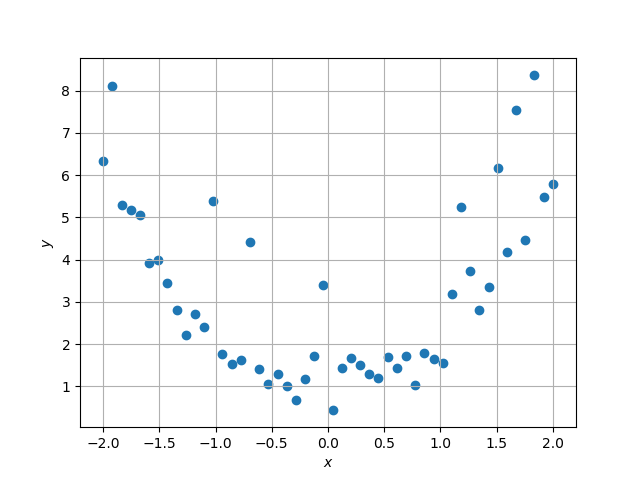
Рассмотрим следующую модель:

,

где ,

и вектор параметров

|  |  |
| --- | --- |
| y | e |
| 6,326918 | 0,546918 |
| 8,103746 | 2,70677 |
| 5,294325 | 0,264379 |
| 5,16982 | 0,490912 |
| 5,047408 | 0,703543 |
| 3,912458 | -0,11236 |
| 3,989166 | 0,267408 |
| 3,453079 | 0,018385 |
| 2,817002 | -0,34662 |
| 2,211339 | -0,69721 |
| 2,712218 | 0,042755 |
| 2,409841 | -0,03653 |
| 5,397173 | 3,157898 |
| 1,7762 | -0,27197 |
| 1,538186 | -0,33487 |
| 1,614139 | -0,09981 |
| 4,427966 | 2,857146 |
| 1,414946 | -0,02874 |
| 1,062119 | -0,27043 |
| 1,295012 | 0,057603 |
| 1,014821 | -0,14344 |
| 0,689545 | -0,40556 |
| 1,164907 | 0,116968 |
| 1,73085 | 0,714082 |
| 3,407608 | 2,406017 |
| 0,441191 | -0,56122 |
| 1,446067 | 0,42685 |
| 1,681483 | 0,629463 |
| 1,510876 | 0,41006 |
| 1,283163 | 0,117557 |
| 1,19504 | -0,05135 |
| 1,688771 | 0,345605 |
| 1,436279 | -0,01966 |
| 1,720068 | 0,13537 |
| 1,022258 | -0,7072 |
| 1,793459 | -0,09675 |
| 1,656263 | -0,41068 |
| 1,553038 | -0,70665 |
| 3,184624 | 0,716211 |
| 5,24953 | 2,556394 |
| 3,739406 | 0,805553 |
| 2,798793 | -0,39177 |
| 3,350055 | -0,11321 |
| 6,157979 | 2,406017 |
| 4,180201 | 0,12355 |
| 7,535232 | 3,157898 |
| 4,472066 | -0,24195 |
| 8,374955 | 3,308274 |
| 5,474926 | 0,039583 |
| 5,790931 | -0,02907 |



1. **Оценки параметров и результаты их сравнения**

Оценки вектора параметров в зависимости от значений параметра функции потерь:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **с = 100** | **с = 10** | **с = 1** | | **с = 0.5** | **с = 0** | **с = -0.5** | **МНК** |
| 1,171494 | 1,070863 | 1,004184 | | 0,994924 | 0,983399 | 0,966217 | 1,19596085 |
| 0,093527 | 0,080656 | -0,00102 | | -0,00803 | -0,01567 | -0,02567 | 0,08980734 |
| 1,396805 | 1,369328 | 1,27495 | | 1,267269 | 1,259932 | 1,253756 | 1,39677964 |
|  | | |
| ИМНК | | |
| 0,075119 | | |
| 0,038686 | | |
| 0,005756 | | |
| 0,004876 | | |
| 0,004526 | | |
| 0,005303 | | |
| МНК | | |
| 0.083492 | | |

Наилучшие оценки вектора достигаются при

Значения весов наблюдений на последней итерации:

|  |
| --- |
|  |
| 1,882842 |
| 0,121925 |
| 1,996271 |
| 1,881205 |
| 1,657848 |
| 1,899749 |
| 1,978164 |
| 1,983489 |
| 1,732936 |
| 1,333474 |
| 1,997634 |
| 1,983831 |
| 0,034133 |
| 1,863547 |
| 1,818617 |
| 1,977761 |
| 0,05952 |
| 1,998077 |
| 1,900326 |
| 1,994118 |
| 1,976336 |
| 1,804566 |
| 1,972085 |
| 1,398155 |
| 0,130327 |
| 1,659155 |
| 1,72673 |
| 1,490377 |
| 1,741943 |
| 1,965509 |
| 1,999785 |
| 1,807449 |
| 1,999644 |
| 1,960662 |
| 1,478443 |
| 1,992728 |
| 1,800407 |
| 1,46073 |
| 1,421387 |
| 0,104375 |
| 1,328571 |
| 1,784438 |
| 1,973315 |
| 0,14724 |
| 1,991431 |
| 0,034014 |
| 1,869933 |
| 0,025591 |
| 1,995587 |
| 1,973025 |

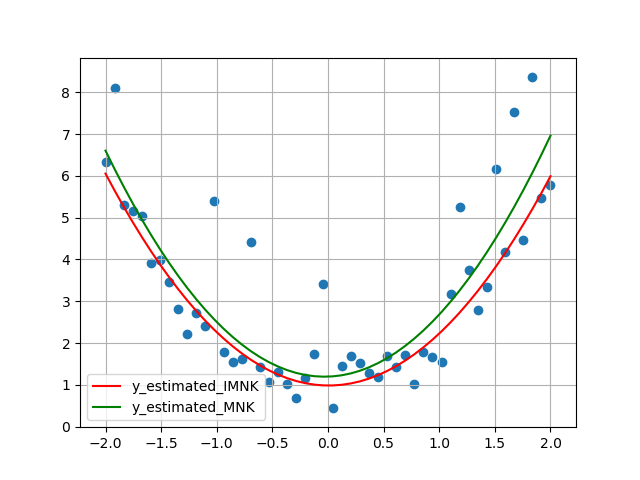
Значения :

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5,78 | 6,054457 | -0,27446 |
| 5,396976 | 5,650168 | -0,25319 |
| 5,029946 | 5,262671 | -0,23273 |
| 4,678908 | 4,891966 | -0,21306 |
| 4,343865 | 4,538053 | -0,19419 |
| 4,024818 | 4,200933 | -0,17611 |
| 3,721758 | 3,880604 | -0,15885 |
| 3,434694 | 3,577068 | -0,14237 |
| 3,163622 | 3,290323 | -0,1267 |
| 2,908549 | 3,020371 | -0,11182 |
| 2,669463 | 2,767211 | -0,09775 |
| 2,446371 | 2,530843 | -0,08447 |
| 2,239275 | 2,311267 | -0,07199 |
| 2,04817 | 2,108483 | -0,06031 |
| 1,873056 | 1,922491 | -0,04943 |
| 1,713949 | 1,753291 | -0,03934 |
| 1,57082 | 1,600884 | -0,03006 |
| 1,443686 | 1,465268 | -0,02158 |
| 1,332549 | 1,346445 | -0,0139 |
| 1,237409 | 1,244413 | -0,007 |
| 1,158261 | 1,159174 | -0,00091 |
| 1,095105 | 1,090727 | 0,004378 |
| 1,047939 | 1,039072 | 0,008867 |
| 1,016768 | 1,004209 | 0,012559 |
| 1,001591 | 0,986138 | 0,015453 |
| 1,002411 | 0,984859 | 0,017552 |
| 1,019217 | 1,000372 | 0,018845 |
| 1,05202 | 1,032678 | 0,019342 |
| 1,100816 | 1,081775 | 0,019041 |
| 1,165606 | 1,147665 | 0,017941 |
| 1,24639 | 1,230346 | 0,016044 |
| 1,343166 | 1,32982 | 0,013346 |
| 1,455939 | 1,446086 | 0,009853 |
| 1,584698 | 1,579144 | 0,005554 |
| 1,729458 | 1,728994 | 0,000464 |
| 1,890209 | 1,895636 | -0,00543 |
| 2,066943 | 2,07907 | -0,01213 |
| 2,259688 | 2,279296 | -0,01961 |
| 2,468413 | 2,496315 | -0,0279 |
| 2,693136 | 2,730125 | -0,03699 |
| 2,933853 | 2,980728 | -0,04687 |
| 3,190563 | 3,248123 | -0,05756 |
| 3,463265 | 3,532309 | -0,06904 |
| 3,751962 | 3,833288 | -0,08133 |
| 4,056651 | 4,151059 | -0,09441 |
| 4,377334 | 4,485622 | -0,10829 |
| 4,714016 | 4,836977 | -0,12296 |
| 5,066681 | 5,205125 | -0,13844 |
| 5,435343 | 5,590064 | -0,15472 |
| 5,820001 | 5,991795 | -0,17179 |

Оценка параметра

1,189539

Сравнение результатов полученных МНК и ИМНК:



**Текст программы**

gendata.py

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import openpyxl

import random

n = 50

x = np.linspace(-2, 2, n)

teta = np.array([1, 0.01, 1.2])

u = np.zeros((n, 1)) #вектор истинных значений отклика

def main():

gen\_response()

def f(x):

f = np.array([1, x, x\*\*2])

return f.T

def gen\_response():

sigma = def\_sigma()

sigma2=np.sqrt((sigma\*\*2)/0.06)

gamma = 8 #количество выбросов

e = np.random.normal(0, sigma, size=(n, 1))

for i in range(gamma + 1):

ind = np.random.randint(n-1, size=1)

e[ind[0], 0]=sigma2\*random.randint(12,22)/10

y = u + e

df1 = pd.DataFrame(y)

df2 = pd.DataFrame(e)

ywriter = pd.ExcelWriter('y.xlsx')

ewriter = pd.ExcelWriter('e.xlsx')

df1.to\_excel(ywriter, 'y', index=False)

df2.to\_excel(ewriter, 'e', index=False)

ywriter.save()

ewriter.save()

f = open('sigma.txt', 'w')

f.write(str(sigma[0, 0]))

f.close()

plt.scatter(x, y)

plt.xlabel(r'$x$')

plt.ylabel(r'$y$')

plt.grid(True)

plt.show()

def def\_sigma():

global u

for xi in range(x.size):

u[xi] = f(x[xi]).T@teta

u\_mean = np.mean(u)

omega2 = ((u - u\_mean).T@(u - u\_mean))/(n-1)

p = 0.06

sigma = np.sqrt(p\*omega2)

return sigma

main()

IMNK.py

import numpy as np

import pandas as pd

from math import pi

from math import sin

from math import exp

import matplotlib.pyplot as plt

n = 50

y = np.zeros(n)

e = np.zeros((n, 1))

x = np.linspace(-2, 2, n)

W = np.zeros((n, n))

X = np.zeros((n, 3))

sigma = 0

sigma\_start = 0

c = 0

def main():

read\_data()

teta\_estimate = np.zeros((3, 6))

quality = np.zeros((6, 1))

teta\_true = np.array([1, 0.01, 1.2])

c\_arr = np.array([100, 10, 1, 0.5, 0, -0.5])

for i in range(c\_arr.size):

global c, sigma

sigma = sigma\_start

c = c\_arr[i]

teta\_estimate[:, i] = IMNK()

quality[i] = (teta\_estimate[:, i] - teta\_true).T@(teta\_estimate[:, i] - teta\_true)

i=quality.argmin()

y\_estimate=np.zeros(n)

y\_estimate = X@teta\_estimate[:, i]

plt.plot(x, y\_estimate, "r-",x,ymnk, "g-")

plt.legend(("y\_estimated\_IMNK","y\_estimated\_MNK"))

plt.scatter(x, y)

plt.grid(True)

plt.show()

df = pd.DataFrame(teta\_estimate)

df1 = pd.DataFrame(quality)

df2 = pd.DataFrame(np.diag(W))

df3 = pd.DataFrame(y\_estimate)

rasn = y.base - e.base - y\_estimate

df4 = pd.DataFrame(rasn)

df5 = pd.DataFrame(sigma)

df6=pd.DataFrame(y.base - e.base)

writer = pd.ExcelWriter('teta\_estimate.xlsx')

df.to\_excel(writer, 'teta\_est', index=False)

df1.to\_excel(writer, 'qua', index=False)

df2.to\_excel(writer, 'w', index=False)

df3.to\_excel(writer, 'y\_est', index=False)

df4.to\_excel(writer, 'y-y\_est', index=False)

df5.to\_excel(writer, 'sigma', index=False)

df6.to\_excel(writer,'y',index=False)

writer.save()

def read\_data():

fy=open("y.txt","r")

fe = open('e.txt', 'r')

fymnk=open("yMNK.txt","r")

yt=[]

ytmnk=[]

et=[]

for i in range(n):

yt.append(float(fy.readline()))

ytmnk.append(float(fymnk.readline()))

et.append(float(fe.readline()))

fy.close

fymnk.close

fe.close

df\_y = pd.DataFrame(yt)

df\_ymnk = pd.DataFrame(ytmnk)

df\_e = pd.DataFrame(et)

global y, e,ymnk

y = df\_y.as\_matrix()

ymnk = df\_ymnk.as\_matrix()

e = df\_e.as\_matrix()

global sigma\_start

f = open('sigma.txt', 'r')

sigma\_start = float(f.read())

f.close()

def f(x):

f = np.array([1, x, x\*\*2])

return f.T

def rk(teta\_estimate\_k):

r = np.zeros((n, 1))

for xi in range(n):

r[xi] = y[xi, 0] - f(x[xi]).T@teta\_estimate\_k

return r

def wk(r):

global W

W = np.zeros((n, n))

for i in range(n):

z = r[i]/sigma

W[i, i] = exp(-z\*z/2)/(1+c\*exp(-z\*z/2))

def build\_X():

for xi in range(x.size):

X[xi, :] = f(x[xi]).T

def stop(teta\_estimate\_k1, teta\_estimate\_k):

el = np.zeros((3, 1))

for i in range(teta\_estimate\_k1.size):

el[i] = abs((teta\_estimate\_k1[i] - teta\_estimate\_k[i])/teta\_estimate\_k[i])

max = el.max()

return max

def sigma\_k(r):

sigma = 0

df\_r = pd.DataFrame(r)

mad = df\_r.mad()

sigma = mad.as\_matrix()

return sigma/0.67449

def IMNK():

# 1

teta\_estimate\_k = np.array([1, 0.01, 1.2])

build\_X()

k = 1

while 1:

# 2

r = rk(teta\_estimate\_k)

wk(r)

# 3

k += 1

teta\_estimate\_k1 = np.linalg.inv(X.T @ W @ X) @ X.T @ W @ y

global sigma

sigma = sigma\_k(r)

# 4

max = stop(teta\_estimate\_k1, teta\_estimate\_k)

if max < 0.0001:

return teta\_estimate\_k1.T

else:

teta\_estimate\_k = teta\_estimate\_k1

main()