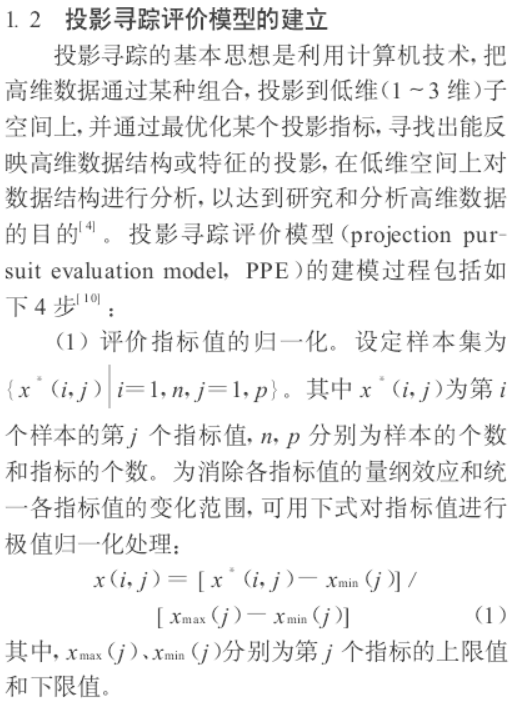
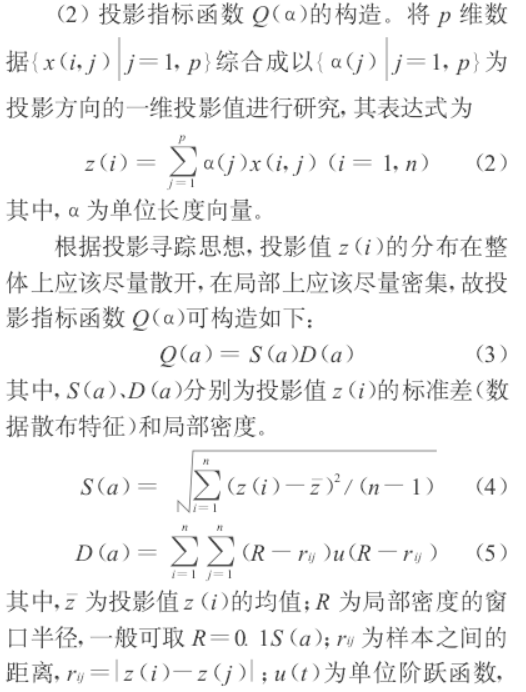
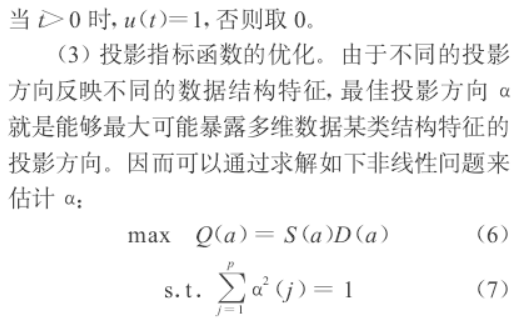
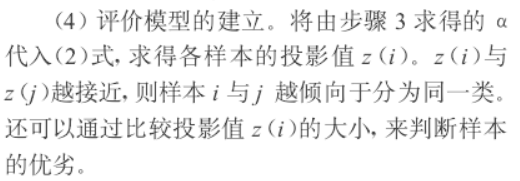
一、PPC投影寻踪评估模型介绍

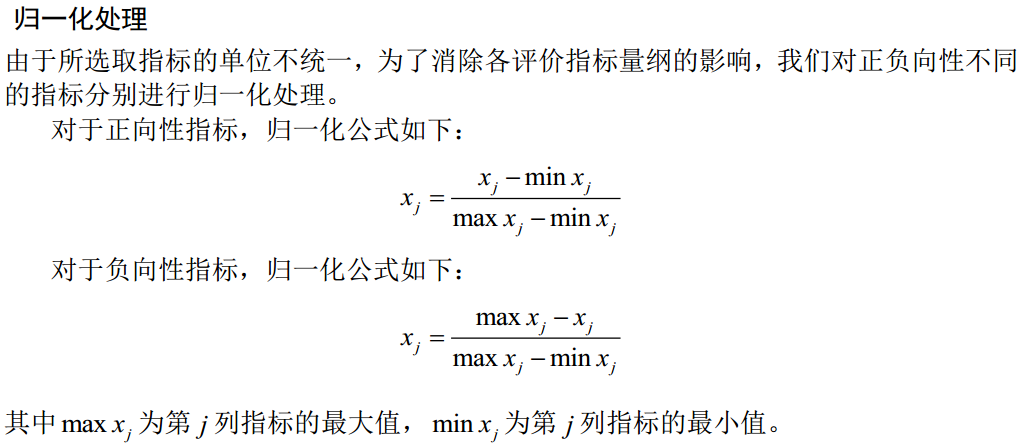


-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

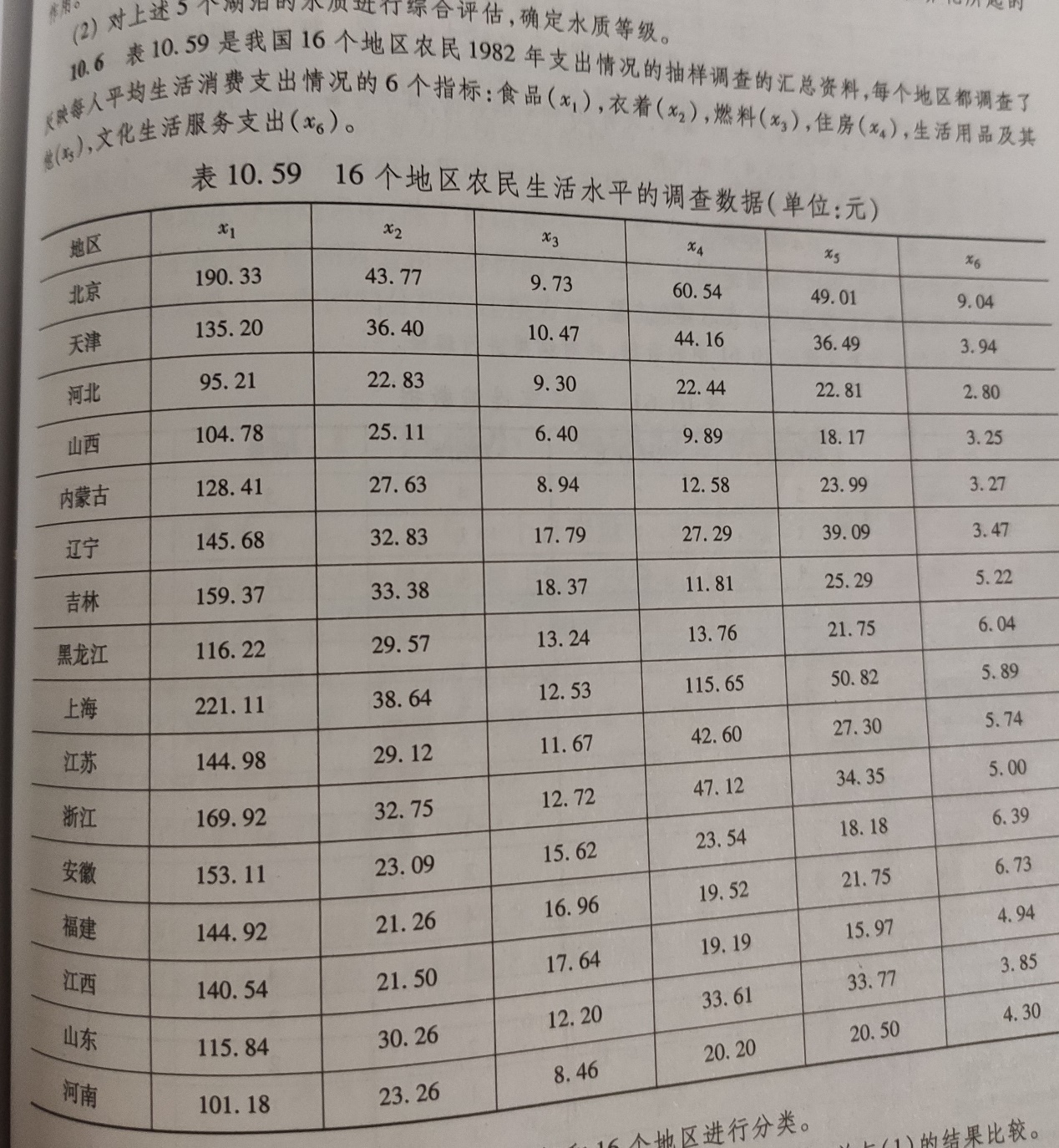


-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-





二、例子：利用PPC评估16个地区消费水平



2.1 Python Code

import numpy as np

from numpy import exp, array, std, sqrt

from numpy.random import rand

def open\_txt(file:str, func=float):

data = []

f = open(file=file, mode='r', encoding='UTF-8')

for line in f:

tmp = line.split()

for i, v in enumerate(tmp):

tmp[i] = func(tmp[i])

data.append(tmp)

return array(data)

data = open\_txt("consume.txt")

n, m = len(data), len(data[0])

# standardization

maxmin = [[data[0][j], data[0][j]] for j in range(m)]

for i in range(n):

for j in range(m):

maxmin[j][0] = max(maxmin[j][0], data[i][j])

maxmin[j][1] = min(maxmin[j][1], data[i][j])

for i in range(n):

for j in range(m):

data[i][j] = (data[i][j] - maxmin[j][1]) / (maxmin[j][0] - maxmin[j][1])

def f(x):

global data, n, m

z = [0 for \_ in range(n)]

for i in range(n):

tmp = 0

for j in range(m):

tmp += (x[j] \* data[i][j])

z[i] = tmp

S = std(z)

R = 0.1 \* S

D = 0

for i in range(n):

for j in range(m):

r = abs(z[i] - z[j])

t, u = R - r, 1

if t <= 0:

u = 0

D += (t \* u)

return S \* D

def get\_new\_x(x:np.array, step):

m = len(x)

new = rand(m)

tmp = sum(new \*\* 2)

new = sqrt(new \*\* 2 / tmp)

return new

new = x + step \* rand(m)

tmp = new \*\* 2

new = sqrt(tmp / sum(tmp))

return new

def SA():

global n, m, data

T, finalT, coef = 1000, 1, 0.97

K, step, niter = 1, 1, 100

tmp = rand(m)

x0 = x = ansX = sqrt((tmp \*\* 2) / sum(tmp \*\* 2))

y0, y, ansY = f(x0), f(x), f(ansX)

while T > finalT:

for \_ in range(niter):

newx = get\_new\_x(x, step)

newy = f(newx)

df1, df2 = newy - y, newy - ansY

if df1 > 0:

x, y = newx, newy

elif exp(df1 / (K \* T)) > rand():

x, y = newx, newy

if df2 > 0:

ansX, ansY = newx, newy

T \*= coef

#print(x0, y0)

print(ansX, ansY)

return ansX, ansY, x0, y0

def evaluate(x):

global data, n, m

z = []

for i in range(n):

tmp = 0

for j in range(m):

tmp += (data[i][j] \* x[j])

z.append(tmp)

print(z)

return z

#from datetime import datetime

#s = datetime.now()

x, y, x0, y0 = SA()

#print("-------------初始状态评估-------------")

#z0 = evaluate(x0)

print("-------------投影寻踪评估-------------")

z = evaluate(x)

#e = datetime.now()

#print("Running Time:", e - s)

province = ['北京', '天津', '河北', '山西', '内蒙古', '辽宁', '吉林', '黑龙江', '上海', '江苏', '浙江', '安徽', '福建', '江西', '山东', '河南']

res = []

for i in range(n):

res.append([province[i], z[i]])

res.sort(key=lambda x:x[1], reverse=True)

print('-------------消费水平由高到低-------------')

print(res)

'''

[0.50526319 0.32080724 0.1190597 0.33010081 0.44054848 0.56970791] 0.31253574570640813

-------------投影寻踪评估-------------

[1.8811314385096543, 0.8871869429853091, 0.17685785455327394, 0.16217123398791233, 0.4019762874941192, 0.8884765746708561, 0.8940325372108173, 0.6517394289551314, 1.8666952338699188, 0.8779154738696757, 1.0758501078324756, 0.7484590084640912, 0.7664622116587704, 0.5215459911643342, 0.6636626651327545, 0.299346035643722]

-------------消费水平由高到低-------------

[['北京', 1.8811314385096543], ['上海', 1.8666952338699188], ['浙江', 1.0758501078324756], ['吉林', 0.8940325372108173], ['辽宁', 0.8884765746708561], ['天津', 0.8871869429853091], ['江苏', 0.8779154738696757], ['福建', 0.7664622116587704], ['安徽', 0.7484590084640912], ['山东', 0.6636626651327545], ['黑龙江', 0.6517394289551314], ['江西', 0.5215459911643342], ['内蒙古', 0.4019762874941192], ['河南', 0.299346035643722], ['河北', 0.17685785455327394], ['山西', 0.16217123398791233]]

'''