

**《数学建模基础创新实践1》**

**实验大作业**

**题目：为亚运代言公司的某商品定价问题**

**学院：卓越学院**

**队员(学号姓名):**

**1.\_\_\_\_\_\_\_22330421干依晗\_\_\_\_\_\_\_\_**

**2.\_\_\_\_\_\_\_22050626王文杰\_\_\_\_\_\_\_**

**3.\_\_\_\_\_\_\_22180105徐堰涛\_\_\_\_\_\_\_\_**

**基于线性回归模型亚运商品定价分析**

**摘要**

杭州2022第19届亚运会申办委员会在筹办过程中实施市场开发计划，某公司为竞标相关代言对某商品做出一次评估。针对在盈利最大情况下确定商品定价这一问题，拟合近年数据得到相关参数，构建了常微分方程模型中的线性回归模型，利用最小二乘法，确定不同情景下的最高定价。该问题的研究能优化代言方案，提高利润增加企业实力及其对亚运会的支持。

针对问题一，经过简单的散点图构造，我们观察引入**线性回归模型**，将近年来的销售情况代入计算，利用最小二乘法拟合求解得出售价一次项系数a，常数项为b ，最终我们得到a=-0.2059，b=13.15，此种情况下拟合得出直线z1=-0.2059x+13.15 ，可见拟合程度较好，能较为准确的反映售价和销售量的关系，可确定销售总收入不低于最初收入时该商品的最高定价为38.91元，销售总收入达到最高时商品定价为31.93元，为确保模型的稳定性，接下来展开对一次项系数a的**灵敏度**分析。

针对问题二，因疫情公司实行技术革新和宣传创新，在固定宣传费50万元的基础上增加了拟投入技术革新费y1，补充浮动宣传费y2，在问题一销量和售价的关系及销售总收入的基础上，构建线性回归模型，结合拟投入技术革新费用和商品价格平方与浮动宣传费与商品价格的关系，将得出的之后总盈利c2与问题一背景下的盈利c1比较，构建动态模型，结合实际情况进行分析，发现取值范围不合理，因此通过迭代计算出合理的范围并对图像进行修正，得到所需要超过的销量和对应定价的关系为x\*z2-0.16x2-0.2x+50≥x(ax+b) 以及取等时的图像为图9。结合图8，可得极值点与交点x坐标的关系，得出在合理范围内可以获得最大盈利和不能获得最大盈利的情况，并计算得到最大盈利可能性η=2.4%，为确保模型的稳定性，接下来展开对k1，k2进行灵敏度分析。

本文的特点在于对问题一进行了**误差分析**，提高了模型的准确性，更符合实际情况，并在问题二中创造性的运用方程的性质，分析最大盈利的可能性。

**关键词：** 线性回归模型 动态模型 误差分析 灵敏度

# 问题重述

## 问题背景

杭州成功取得亚运会举办权，绵延中国红的精神力量，在充盈杭州精神文化的同时，带动了杭州各行各业和各公司的共同发展，增强了社会活力。

杭州2022年第19届亚运会的申办委员会为成功举办杭州亚运会，提高杭州亚运会的品牌价值，增强社会对杭州亚运会的关注度和参与度实施市场开发计划。某公司抓住亚运契机，扩大某商品影响力，增加利润，因此该公司事先对这一进军亚运的商品做出事先评估，在更大盈利的条件下给商品定价。

## 1.2问题重述

1.已知该商品初售价25元/件时，年销售量为8万件，根据近年商品售价与销量统计数据分析关系，构建数学模型，当销售总收入不低于最初收入时确定商品最高定价，并求出商品总收入最高时的商品定价。

2. 受疫情影响，公司启动一轮创新，在固定宣传费50万元基础上，已知拟投入技术革新费用（单位：万元）的平方呈y1=k1x2+b（k1=0.16，b=-100）的线性关系，补充浮动宣传费（单位：万元）和商品价格呈y2=k2x(k2=0.2)的正比例关系，构建数学模型，确定销量超出多少件能比问题一的研究更盈利，确定此时定价，并分析获利最大的可能性。

# 问题分析

本文要解决商品如何盈利最大的问题，问题一要求只从定价和销售额的关系，求得收入达到最高时的定价，问题二在进行宣传和技术革新的情况下，提高盈利，并分析获得最大盈利的可能性。

## 问题一的分析

问题一中给出了商品初售价和年销量的数据，由于商品售价和年销量的关系相对稳定，通过数据分析散点图可以进一步确认售价对销量存在着线性阻滞关系。由此，可以考虑构建常微分方程模型中的线性回归模型对销售总收入进行分析。

## 问题二的分析

问题二中设定了宣传和技术革新费用，考虑两者与售价分别存在线性关系和非线性关系，选择线性回归模型刻画最终盈利与售价的关系，并比较其与问题一背景下的盈利情况，进一步选择使用动态模型得到所需要超过的销量和对应定价的关系。

# 模型假设

1. 假设问题一背景下售价和销量保持线性关系
2. 假设商品销售所得收入减去技术革新费用和宣传费用即为盈利
3. 假设定价只考虑销售总收入、盈利的影响
4. 假设不存在经济危机、生产事故和其他同类商品竞争影响销售量

# 符号说明

表1 符号系统

|  |  |  |
| --- | --- | --- |
| 变量 | 变量描述 | 单位 |
| *x* | 售价 | 元 |
| *z1* | 之前销量 | 万件 |
| *c1* | 销量总收入 | 万元 |
| *xmax* | 最高定价 | 元 |
| *x0* | 总收入最高时定价 | 元 |
| *y* | 总投入 | 万元 |
| *z2* | 后来销量 | 万件 |
| *zq* | 合理销量左边界 | 万元 |
| *zh* | 合理销量右边界 | 万元 |
| *c2* | 总盈利 | 万元 |
| *η* | 最大盈利可能性 |  |
| *xi* | 交点横坐标 | 元 |
| *xj* | 对称轴横坐标 | 元 |
| *z0* | 可否取得最大盈利的临界值 | 万元 |

# 模型的建立与求解

## 问题一的建模与求解

### 模型准备

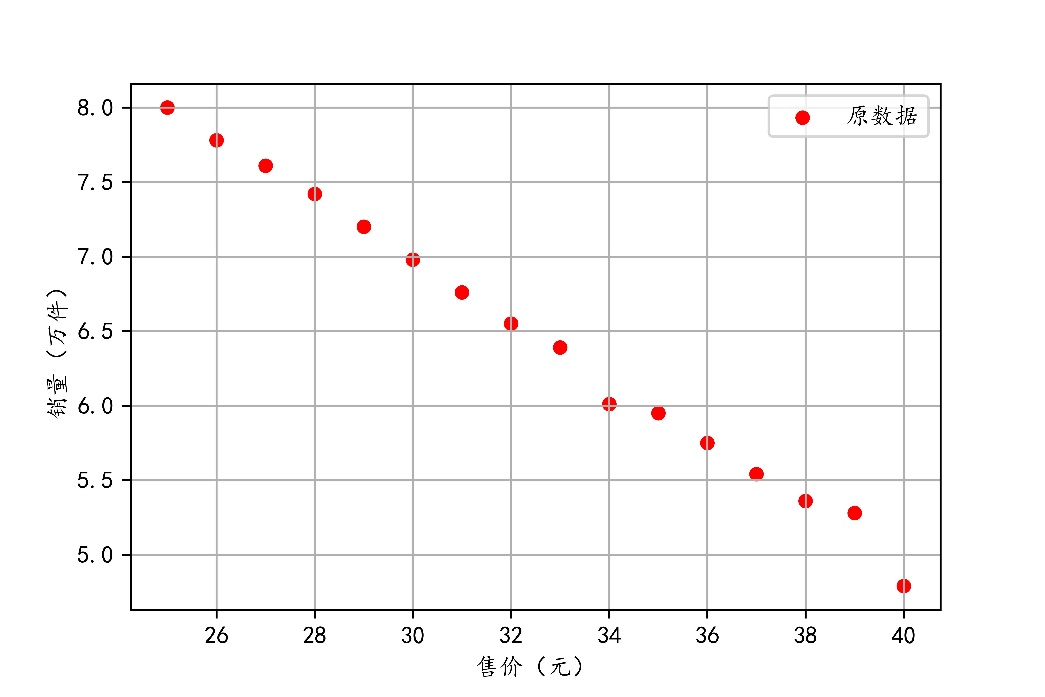
 我们利用python软件绘制销售量与售价之间的散点图如下：

图1 商品售价和销量关系散点图

分析以上散点图可以得知，商品销售量与售价符合线性关系，在此我们可以使用一次函数拟合数据并进一步建立线性回归模型。

### 线性回归模型的建立

假设售价与销量的关系为

z1=ax+b (1)

其中a为一次项系数，b为常数项，x为售价，z1为销量

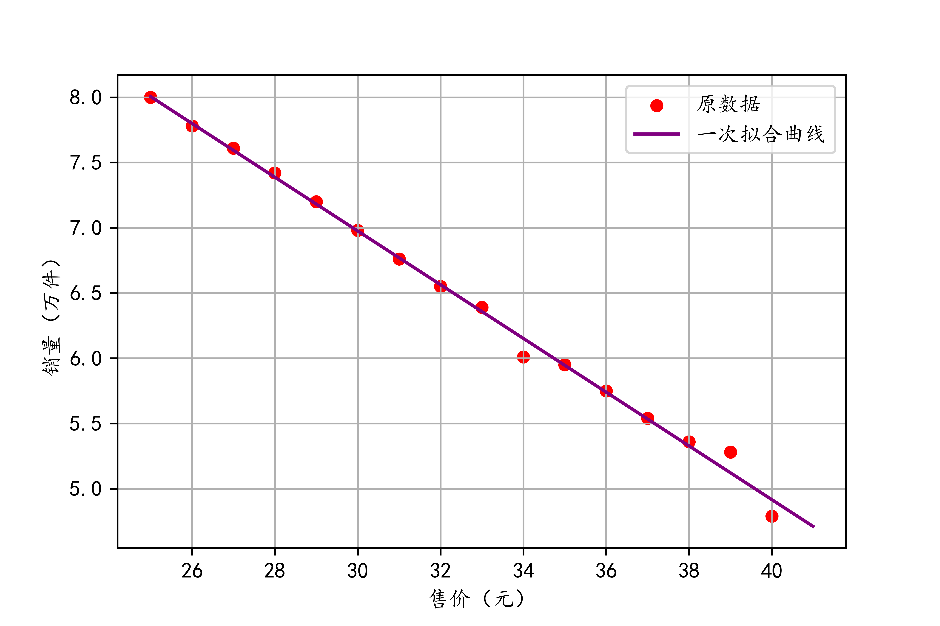
使用python根据最小二乘法通过多项式拟合得到：

图2 商品售价和销售量拟合曲线

求得a、b的估算值为：

a= -0.2059 b= 13.15 (2)

将（2）代入（1），所得售价和销量关系为

z1=-0.2059x+13.15 (3)

假设销售总收入为c1，结合（3）可得

c1=（-0.2059x+13.15）x (4)

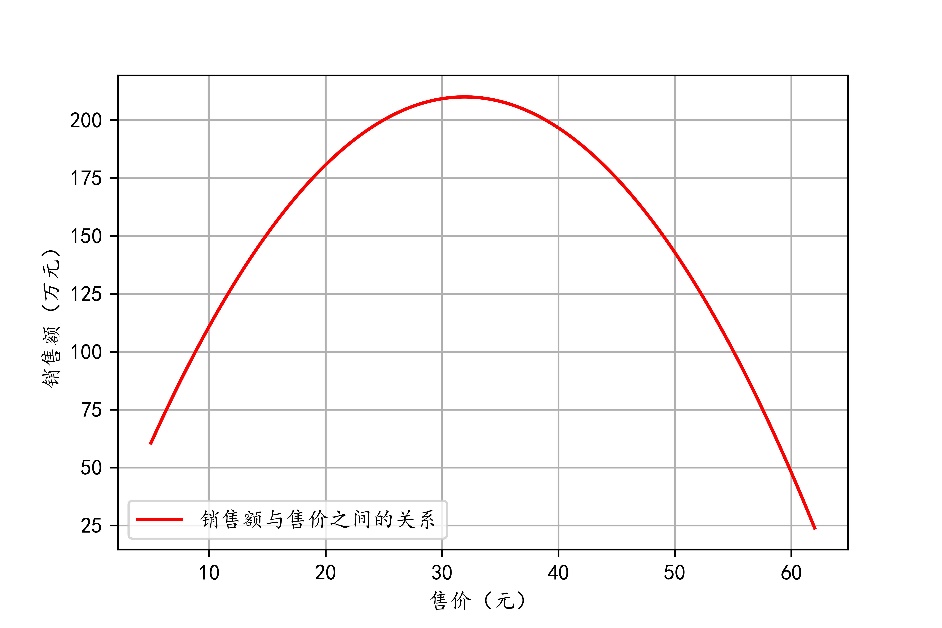
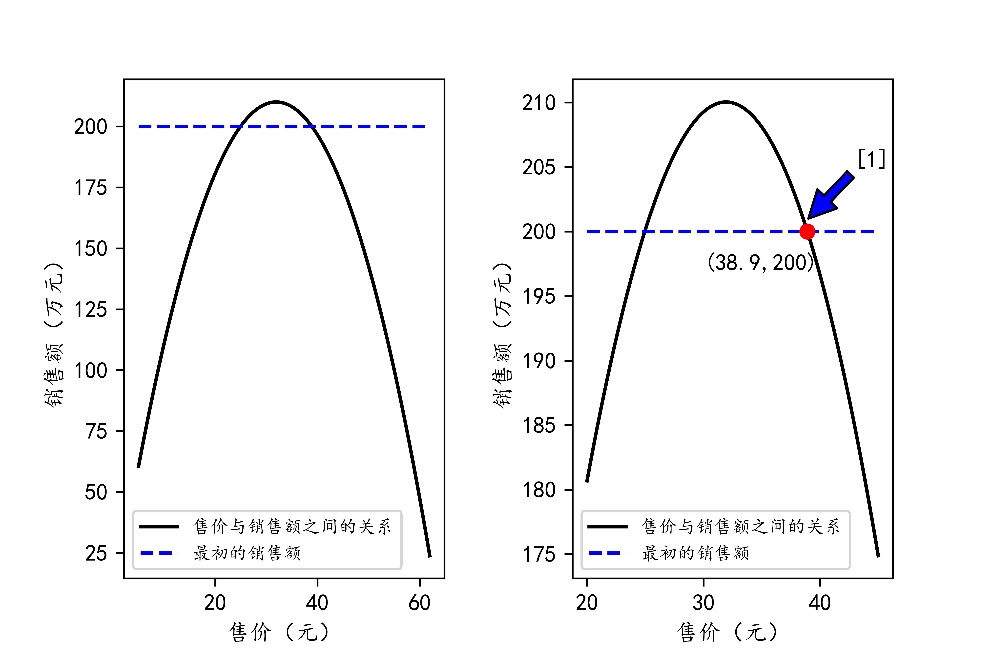
 绘制图像如下：

图3 销售额与售价的关系

### 线性回归模型的求解

已知最初销售额为200万元，需确定销售总收入不低于最初销售额的取值范围

图4 对比图

为便于观察，放大左图得右图，分析右图可知，满足条件的最高定价为[1]

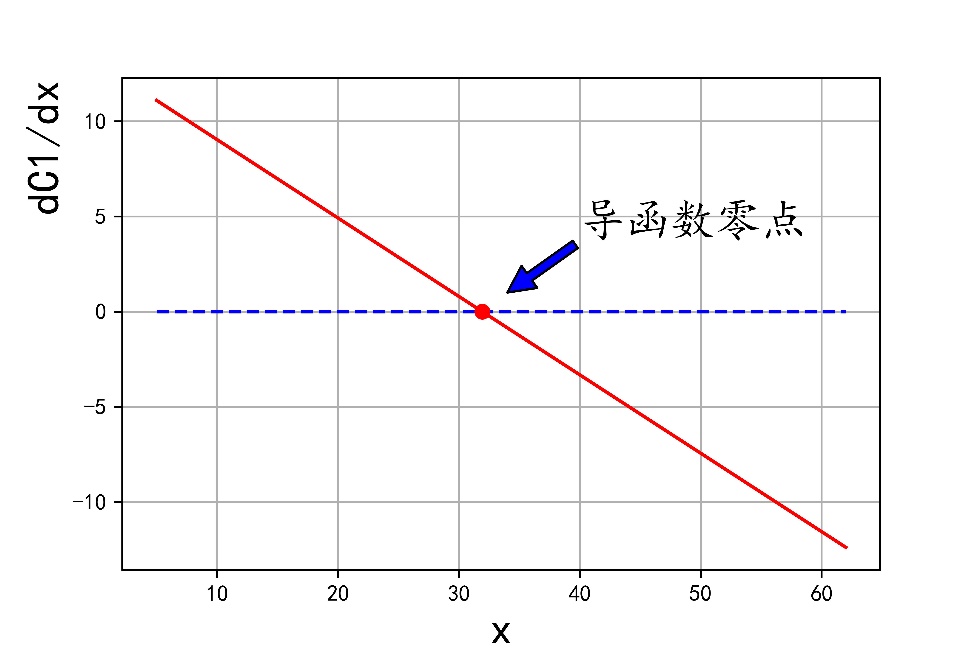
为进一步确定商品的销售总收入达到最高值的定价，通过销售额对售价取微分得到销售额关于售价的导函数，该函数零点即为销售总收入达到最高时的商品定价，如下：

图5 销售额关于售价的导函数图

### 结果分析

根据附录一，求得商品的最高定价：

xmax=38.91 (5)

根据附录一，当销售总收入达到最高时，商品定价为：

x0= 31.93 (6)

### 模型检验

根据（4），可计算x=25，30，35时的销售额：

当x=25时

c1（25）=（-0.2059\*25+13.15）\*25=200.1 (7)

当x=30时

c1（30）=（-0.2059\*30+13.15）\*30=209.3 (8)

当x=35时

c1（35）=（-0.2059\*35+13.15）\*35=208.1 (9)

其相对误差分别为

=0.05% (10)

=0.048% (11)

=0.072% (12)

观察数据发现，误差较小，说明模型较符合实际情况，从而证明计算结果的准确性。

## 问题二的建模与求解

### 模型准备

假设c2为公司启动技术革新和宣传时的总盈利，z2为此时的销售量，y为投入的总宣传费和技术革新费的总和，结合问题一的思路，构建线性回归模型，可得：

c2=x\*z2-y1-y2-50 (13)

由题意可得，拟投入技术革新费用跟商品价格平方的关系为：

y1=k1x2+b (14)

已知k1=0.16，b=-100

补充浮动宣传费和商品价格的关系为：

y2=k2x (15)

已知k2=0.2

因此时较问题一背景下的情况更盈利，即在售价相同的情况下：

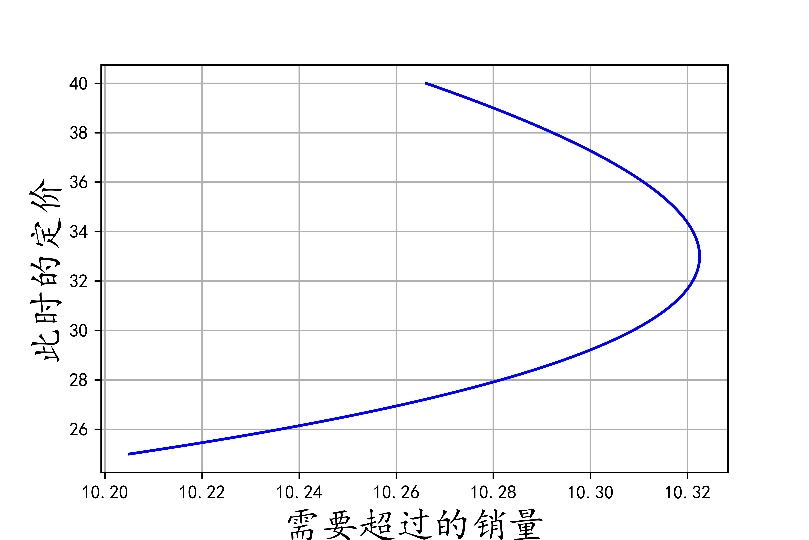
c2＞c1 (16)

### 模型建立

结合（3）（4）（13）（14）（15）可得:

x\*z2-0.16x2-0.2x+50≥x(ax+b) (17)

分析可得，（17）取等时为临界条件，据此画图得：

图6 超过的销量和定价的关系图

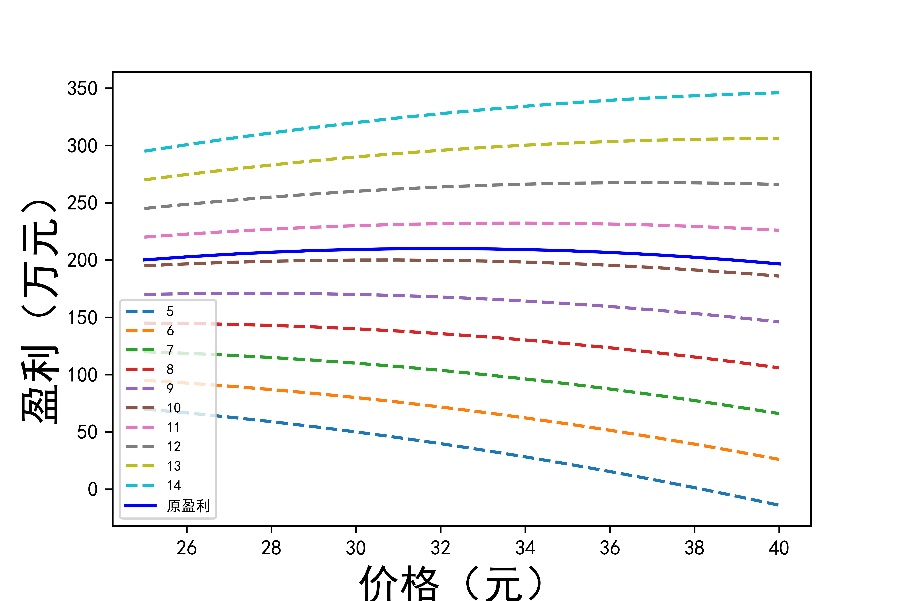
观察图像发现，图像中存在不合理区间，结合实际情况分析，假设销售量极少时盈利难度极大，反之则较易，在两种极端下模型与事实不符，故为提高模型的准确性，需将销量控制在合理范围内考虑问题，根据（4）（13）构建动态模型并令整数z2∈[5,14]绘制图像,如下：

图7 价格和盈利关系图

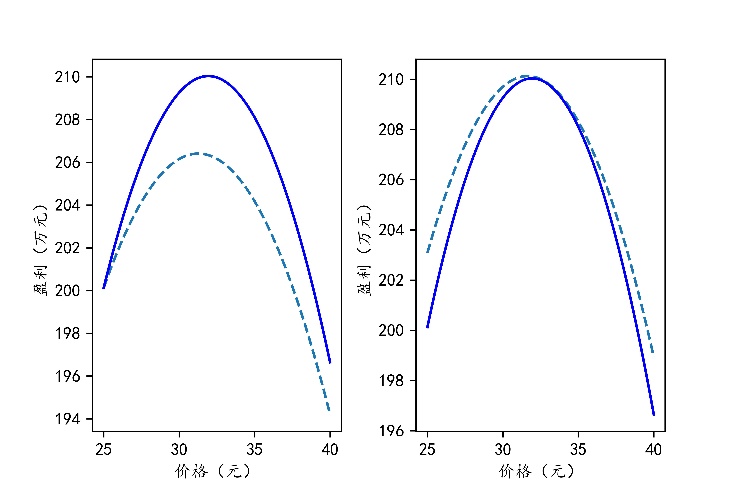
观察图像可知在销量取10-11万之间，计算得出销量的合理范围，绘制图像如下：

图8 合理范围的两个临界图

### 模型求解及结果分析

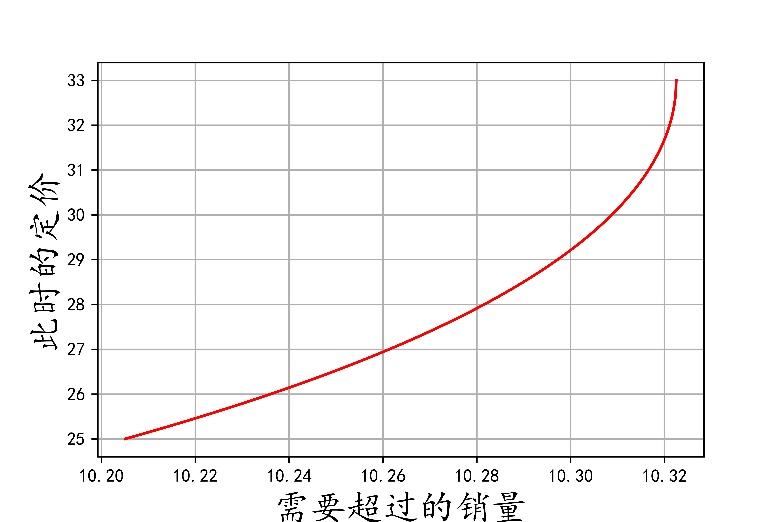
根据附录2，反复迭代求得，此时的销量分别为zq=10.21万件，zq=10.32万件，据此分析，销量低于10.21万件时不可能比之前更加盈利，当其超过10.32万件时任何合理定价都会更加盈利，由此将图6修正到合理范围，绘制图像如下：

图9 对图6的修正图

结合（17）和图9，可以确定所需要超过的销量和此时的商品定价。

结合图8，进一步分析最大盈利的可能性。在图8的合理范围内，两图像存在唯一交点，在交点左侧满足比之前更加盈利，而在交点右侧不满足更加盈利，因此问题的可行域为交点左侧的部分；最大盈利表现为函数极值点，根据二次函数的性质，函数的极值点在对称轴处取得。为求得最大盈利的可能性，需比较不同销售量下的对称轴和极值点的大小。

联立(4)(13)式，设交点横坐标为xi：

c1=（-0.2059x+13.15）x (18)

c2=x\*z2-y1-y2-50

设c2关于x的函数图像对称轴为xj，比较xi和xj得：

xi-xj (19)

分析可得，(19)式是关于z2的函数，设其零点为z0，当(19)式不小于零时，即z2∈[z0 ,zh]可以取得最大盈利；当(19)式小于零时，即z2∈[zq ,z0]不能取得最大盈利。结合合理范围，设η为最大盈利可能性。

η= \*100% (20)

根据附录2可知：

η=2.4% (21)

综上所述，

(22)

# 灵敏度分析

为了进一步确定模型的稳定性，我们对问题一、二分别进行灵敏度分析。

## 问题一分析

a的变化会对模型结果产生影响，因此对系数a进行灵敏度分析。

首先假设x=25不变，

S(c1,a)=25\*25\* (22)

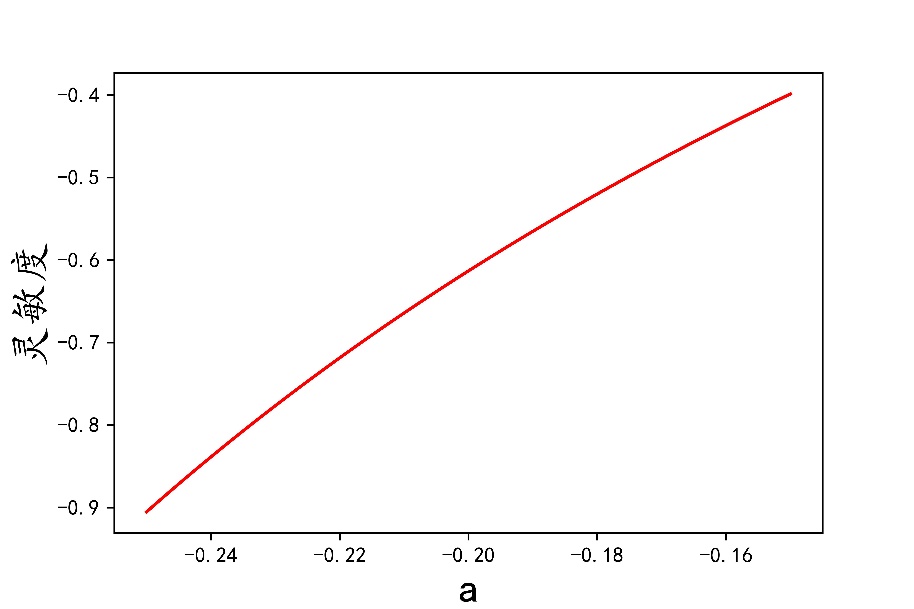
 由此可得，

图10 参数a灵敏度的分析图

其中，当a=-0.2059时，灵敏度为-0.61，观察图像可知，灵敏度普遍较小，模型较稳定。

## 问题二的分析

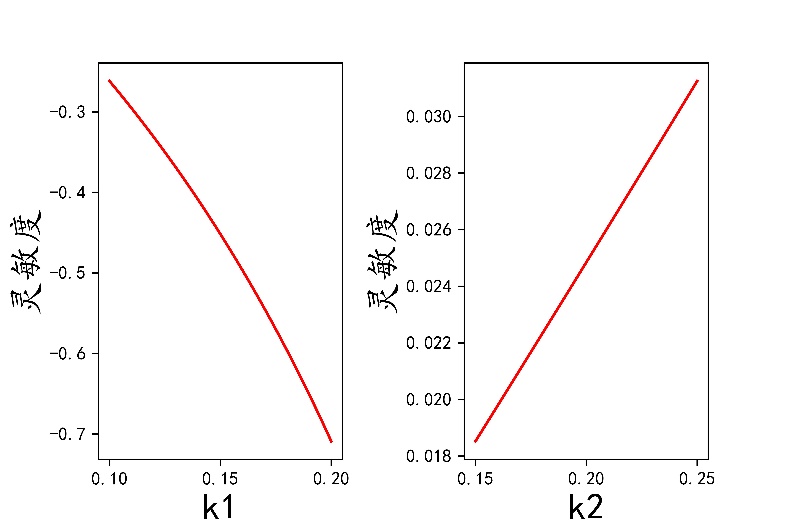
k1 , k2的变化会对模型结果产生影响，因此对其进行灵敏度分析。

再次假设x=25，y=10.25不变，

S(c2,k1)= (23)

S(c2,k2)= (24)

由此可得，

图11 k1，k2灵敏度分析图

其中当k1=0.16时，灵敏度为-0.49，当k2=0.2时，灵敏度为0.024，观察图像可知，灵敏度普遍较小，模型较稳定。

通过一系列分析，发现模型关于系数的灵敏度极小，模型较稳定。

# 模型的评价与改进

## 模型的优点

1. 图像的直观性：解题过程伴随图像展示，更为客观、直接的展现本组解题思路。
2. 建模的科学性：本文基于线性回归模型和动态模型，与已知数据拟合，得到了较稳定的模型和准确的结果。
3. 求解的严谨性：求解过程注重误差和灵敏度分析，并最大程度与实际相结合。

## 模型的缺点与改进

1. 问题一假设过于理想，对销量与售价关系的稳定性要求较高，在研究盈利问题时忽略了商品成本这一因素，并忽视较多现实因素产生的影响。
2. 模型仅适用于题目的局限背景下，应当考虑更多现实因素优化模型，使模型更具实际意义。

# 附录

|  |  |  |  |
| --- | --- | --- | --- |
| 附录名称 | 问题一 | 代码语言 | Python |
| %一次拟合曲线  import numpy as np  import matplotlib.pyplot as plt  from scipy import optimize  import pylab  def func(x, a, b): # 需要拟合的函数  return a\*x+b  # 拟合点  plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  x = [i for i in range(25, 41, 1)]#售价  y = [8.00,7.78,7.61,7.42,7.20,6.98,6.76,6.55,6.39,6.01,5.95,5.75,5.54,5.36,5.28,4.79]#销量  a1, b1 = optimize.curve\_fit(func, x, y)[0]  x1 = np.arange(25, 41, 0.01)  y1 = a1\*x1+b1  print(x1.shape,y1.shape)  print(a1,b1)  z=func(35,a1,b1)  print(z)  plt.scatter(x[:], y[:], 25, "red")  plt.grid()  plt.plot(x1, y1, "purple")  plt.legend(['原数据','一次拟合曲线'])  plt.xlabel('售价（元）')  plt.ylabel('销量（万件）')  plt.savefig('售价和销量的关系.png', dpi=1000)  plt.show()  %二次拟合曲线  import math  import numpy as np  import matplotlib.pyplot as plt  from scipy import optimize  import pylab  import scipy  def func(x): # 需要拟合的函数  return (-0.2059264703592322\*x+13.153235286675047)\*x  # 拟合点  a = -0.2059264703592322  b = 13.153235286675047  x0 = (-b-math.sqrt(b\*b-4\*a\*(-200)))/(2\*a)  plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  plt.subplot(121)  x = np.linspace(5,62,1000)#售价  R0 = np.ones(1000)  R0 = R0\*200  R = func(x)  plt.xlabel('售价（元）')  plt.ylabel('销售额（万元）')  plt.plot(x,R,'k-',x,R0,'b--')  plt.legend(['销售额与售价之间的关系','最初的销售额'],fontsize=8)  print(x0)  print((func(x0)))  plt.subplot(122)  x = np.linspace(20,45,1000)#售价  R0 = np.ones(1000)  R0 = R0\*200  R = func(x)  plt.annotate('[1]',fontsize=10,xy=(39,201),xytext=(43,205),arrowprops=dict(facecolor='blue',edgecolor='black'))  plt.text(30,197,'(38.9,200)',fontsize=10)  plt.plot(x,R,'k-',x,R0,'b--',x0,200,'ro')  plt.xlabel('售价（元）')  plt.ylabel('销售额（万元）')  plt.legend(['销售额与售价之间的关系','最初的销售额'],fontsize=8)  plt.subplots\_adjust(wspace =0.4)#调整子图间距  plt.savefig('销售额和售价的关系.jpg', dpi=1000)  plt.show()  %导函数曲线  import math  import sympy as sp  import numpy as np  import matplotlib.pyplot as plt  from scipy import optimize  import pylab  import scipy  import sympy as sp  x = sp.Symbol('x')  f = -0.2059264703592322\*2\*x+13.153235286675047  x = sp.solve(f)  x0=x  print(x)  def func(x): # 需要拟合的函数  return 2\*(-0.2059264703592322)\*x+13.153235286675047  plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  x = np.linspace(5,62,1000)  y = func(x)  y0 = np.zeros(1000)  plt.plot(x,y,'r-',x,y0,'b--',x0,0,'ro')  plt.xlabel('x',fontsize=20)  plt.ylabel('dC1/dx',fontsize=20,loc='top')  plt.grid()  plt.annotate('导函数零点',fontsize=20,xy=(34,1),xytext=(40,4),arrowprops=dict(facecolor='blue',edgecolor='black'))  plt.savefig('关于销售额的导函数图像.jpg',dpi=1000)  plt.show()  %图10  import math  import numpy as np  import matplotlib.pyplot as plt  from scipy import optimize  import pylab  import scipy  import sympy  from scipy.optimize import fsolve  from mpl\_toolkits.mplot3d import Axes3D  from sympy import symbols, diff  from tqdm import tqdm  a = np.linspace(-0.15,-0.25,1000)  k1 = np.linspace(0.1,0.2,1000)  k2 = np.linspace(0.15,0.25,1000)  fig = plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  b = 13.153235286675047  def func1(a):  return 25\*25\*a/(25\*(25\*a+b))  plt.xlabel('a',fontsize=20)  plt.ylabel('灵敏度',fontsize=20)  print(func1(-0.2))  plt.plot(a,func1(a),'r-')  plt.savefig('a灵敏度.jpg',dpi=1000)  plt.show() | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 附录名称 | 问题二 | 代码语言 | Python |
| %图6  def func1(x):  return (0.16+a) \* x + (-100)/x + (0.2+b) + 50/x  x = np.linspace(25,40,1000)  z2 = func1(xl)  plt.xlabel('需要超过的销量',fontsize=20)  plt.ylabel('此时的定价',fontsize=20)  plt.plot(z2,x,'b-')  plt.grid()  plt.savefig('需要超过的销量与此时的定价关系.jpg',dpi=1000)  plt.show()  %图7  import matplotlib.pyplot as plt  #设销售量与售价的关系为未知，建立P的表达式  def func(x,y):  return x \* y - 0.16 \* x \*\* 2 - (-100) - 0.2 \* x - 50  #对比原来的销售量与售价的关系对应的P  def func0(x):  return (-0.2059264703592322\*x+13.153235286675047)\*x  for yi in range(5,15):  P = func(x,yi)  plt.plot(x,P,'--',label=yi)  P0 = func0(x)  plt.plot(x,P0,'b-',label='原盈利')  plt.xlabel('价格（元）',fontsize=20)  plt.legend(loc='lower left',fontsize=7)  plt.ylabel('盈利（万元）',fontsize=20)  plt.savefig('遍历图.jpg',dpi=1000)  plt.show()  %图8  fig = plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  a = -0.2059264703592322  b = 13.153235286675047  x = np.linspace(25,40,1000)  y0 = a\*x+b  #设销售量与售价的关系为未知，建立P的表达式  def func(x,y):  return x \* y - 0.16 \* x \*\* 2 - (-100) - 0.2 \* x - 50  #对比原来的销售量与售价的关系对应的P  def func0(x):  return (-0.2059264703592322\*x+13.153235286675047)\*x  for yi in range(5,15):  P = func(x,yi)  plt.plot(x,P,'--')  P0 = func0(x)  plt.plot(x,P0,'b-')  plt.xlabel('价格（元）')  plt.ylabel('盈利（万元）')  #由图可知原来的关系在yi对应10--11,先粗略求出一直大于原函数保持盈利的近似临界位置  tr = np.linspace(10,11,1000)  sign = 1  for i in tr:  P = func(x,i)  k = 0  for j in range(0,1000) :  if P[j]>=P0[j] :  k+=1  if k==1000 :  print("i1 = ",i )  sign=0  if sign==0 :  break  plt.subplot(122)  ym = func(x,i)  plt.plot(x,ym,'--')  plt.plot(x,P0,'b-')  plt.xlabel('价格（元）')  plt.ylabel('盈利（万元）')  '''plt.legend([])'''  tr = np.linspace(10,11,1000)  n = 1  t = 0  temp = []  for i in tr :  P = func(x,i)  k = 0  sign = 0  for j in range(0,1000) :  if P[j]>P0[j] :  k+=1  if k>=n :  t = i  temp.append(i)  sign = 1  break  if sign==0 :  temp.append(t)  if temp==1 :  n+=1  temp1 = np.array(temp)  tr = np.linspace(10.10,10.30,1000)  sign = 0  for i in tr :  P = func(x,i)  k=0  for j in range(0,1000) :  if P[j]>=P0[j] :  k+=1  if k==1 :  print("i2 = ",i)  sign = 1  break  if sign ==1 :  break  y1 = func(x,i)  plt.subplot(121)  plt.plot(x,y1,'--')  plt.plot(x,P0,'b-')  plt.xlabel('价格（元）')  plt.ylabel('盈利（万元）')  plt.savefig('临界情况图.jpg',dpi=1000)  plt.subplots\_adjust(wspace=0.3)  plt.show()  %图9  r=[]  tr2=np.linspace(10.205105105105105,10.322512512512512,1000)  for q in tr2:    f6=lambda x:x\*(args[0]\*x+args[1])-(x\*q-0.16\*x\*\*2-(-100)-0.2\*x-50)  a=fsolve(f6,[1])  if a>=25 and a<=40 :  r.append(a)    plt.plot(tr2,r,'r-')  plt.xlabel('需要超过的销量',fontsize=20)  plt.ylabel('此时的定价',fontsize=20)  plt.grid()  plt.savefig('修正图.jpg',dpi=1000)  plt.show()  def func1(x):  return (0.16+a) \* x + (-100)/x + (0.2+b) + 50/x  x = np.linspace(25,40,1000)  z2 = func1(xl)  plt.xlabel('需要超过的销量',fontsize=20)  plt.ylabel('此时的定价',fontsize=20)  plt.plot(z2,x,'b-')  plt.grid()  plt.savefig('需要超过的销量与此时的定价关系.jpg',dpi=1000)  plt.show()  import matplotlib.pyplot as plt  #设销售量与售价的关系为未知，建立P的表达式  def func(x,y):  return x \* y - 0.16 \* x \*\* 2 - (-100) - 0.2 \* x - 50  #对比原来的销售量与售价的关系对应的P  def func0(x):  return (-0.2059264703592322\*x+13.153235286675047)\*x  for yi in range(5,15):  P = func(x,yi)  plt.plot(x,P,'--',label=yi)  P0 = func0(x)  plt.plot(x,P0,'b-',label='原盈利')  plt.xlabel('价格（元）',fontsize=20)  plt.legend(loc='lower left',fontsize=7)  plt.ylabel('盈利（万元）',fontsize=20)  plt.savefig('遍历图.jpg',dpi=1000)  plt.show()  fig = plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  a = -0.2059264703592322  b = 13.153235286675047  x = np.linspace(25,40,1000)  y0 = a\*x+b  #设销售量与售价的关系为未知，建立P的表达式  def func(x,y):  return x \* y - 0.16 \* x \*\* 2 - (-100) - 0.2 \* x - 50  #对比原来的销售量与售价的关系对应的P  def func0(x):  return (-0.2059264703592322\*x+13.153235286675047)\*x  for yi in range(5,15):  P = func(x,yi)  plt.plot(x,P,'--')  P0 = func0(x)  plt.plot(x,P0,'b-')  plt.xlabel('价格（元）')  plt.ylabel('盈利（万元）')  #由图可知原来的关系在yi对应10--11,先粗略求出一直大于原函数保持盈利的近似临界位置  tr = np.linspace(10,11,1000)  sign = 1  for i in tr:  P = func(x,i)  k = 0  for j in range(0,1000) :  if P[j]>=P0[j] :  k+=1  if k==1000 :  print("i1 = ",i )  sign=0  if sign==0 :  break  plt.subplot(122)  ym = func(x,i)  plt.plot(x,ym,'--')  plt.plot(x,P0,'b-')  plt.xlabel('价格（元）')  plt.ylabel('盈利（万元）')  '''plt.legend([])'''  tr = np.linspace(10,11,1000)  n = 1  t = 0  temp = []  for i in tr :  P = func(x,i)  k = 0  sign = 0  for j in range(0,1000) :  if P[j]>P0[j] :  k+=1  if k>=n :  t = i  temp.append(i)  sign = 1  break  if sign==0 :  temp.append(t)  if temp==1 :  n+=1  temp1 = np.array(temp)  tr = np.linspace(10.10,10.30,1000)  sign = 0  for i in tr :  P = func(x,i)  k=0  for j in range(0,1000) :  if P[j]>=P0[j] :  k+=1  if k==1 :  print("i2 = ",i)  sign = 1  break  if sign ==1 :  break  y1 = func(x,i)  plt.subplot(121)  plt.plot(x,y1,'--')  plt.plot(x,P0,'b-')  plt.xlabel('价格（元）')  plt.ylabel('盈利（万元）')  plt.savefig('临界情况图.jpg',dpi=1000)  plt.subplots\_adjust(wspace=0.3)  plt.show() | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 附录名称 | 灵敏度分析 | 代码语言 | Python |
| %问题一  import math  import numpy as np  import matplotlib.pyplot as plt  from scipy import optimize  import pylab  import scipy  import sympy  from scipy.optimize import fsolve  from mpl\_toolkits.mplot3d import Axes3D  from sympy import symbols, diff  from tqdm import tqdm  a = np.linspace(-0.15,-0.25,1000)  k1 = np.linspace(0.1,0.2,1000)  k2 = np.linspace(0.15,0.25,1000)  fig = plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  b = 13.153235286675047  def func1(a):  return 25\*25\*a/(25\*(25\*a+b))  plt.xlabel('a')  plt.ylabel('灵敏度')  print(func1(-0.2))  plt.plot(a,func1(a),'r-')  plt.show()  %问题二  import math  import numpy as np  import matplotlib.pyplot as plt  from scipy import optimize  import pylab  import scipy  import sympy  from scipy.optimize import fsolve  from mpl\_toolkits.mplot3d import Axes3D  from sympy import symbols, diff  from tqdm import tqdm  a = np.linspace(-0.15,-0.25,1000)  k1 = np.linspace(0.1,0.2,1000)  k2 = np.linspace(0.15,0.25,1000)  fig = plt.figure()  plt.rcParams['font.sans-serif'] = ['KaiTi'] # 指定字体 KaiTi（楷体）  plt.rcParams['axes.unicode\_minus'] = False  b = 13.153235286675047  plt.subplot(121)  def func2(k1):  return -625\*k1/(25\*10.25-(625\*k1-100)-5-50)  plt.xlabel('k1')  plt.ylabel('灵敏度')  print(func2(0.16))  plt.plot(k1,func2(k1),'r-')  plt.subplot(122)  def func3(k3):  return 25\*k3/(25\*10.25-25\*k3-50)  plt.xlabel('k2')  plt.ylabel('灵敏度')  print(func3(0.2))  plt.plot(k2,func3(k2),'r-')  plt.subplots\_adjust(wspace=0.5,hspace=0.5)  #a -61%  #k1 -49%  #k2 2.4%  plt.show() | | | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |