# 课题：互联网文件大小的特征分析

## Task1：分析文件大小的概率分布

1. 数据处理

在已经学过的语言中，我们选择了 C 语言来进行数据的处理，即利用 cpp 程序将老师从网站上爬取的数据中提取文件的大小。


程序实现的思路很简单，每更新一次文件，就记录一次文件的大小，之后将数据放进开辟的数组中，依照顺序输出有多少个同样大小的文件。

由于数据过于密集，我们再次将数据进行了进一步的处理，将文件的大小的每十个一组来进行数据的输出。使数据在处理之后能够更加完整的模拟发生在现实中的事情。消除了一定的数据的抖动。同时为了之后的泊松分布的分析图，我们将 file_size 除以 10。

```
while (scanf("%s", s) != EOF)
{
    int l = strlen(s);
    int index = 0;
    bool flag = 0;
    for (int i=0; i<l; ++i)
    {
        if (s[i] == '(')
        {
            for (int j=i+1; j<l&&isdigit(s[j]); ++j)
                ss[index++] = s[j];
            ss[index] = '\0';
            flag = 1;
        }
        if (flag)
        {
            num[total++] = atoi(ss);
            break;
        }
    }
}

 double p;
 for (int i=1; i<=2000; ++i)
 {
     p+=rec[i];
     if (i%10 == 0)
     {
         if (p/(double)total)
         {
             printf("%f %f\n", double(i-5)/10,p/(double)total);
             p = 0;
         }
     }
 }
```

输出的结果部分如下，左边为 file_size/10，右为相应的概率。

```
0.500000  0.027887
1.500000  0.059910
2.500000  0.059554
3.500000  0.044005
4.500000  0.028172
5.500000  0.022680
6.500000  0.021111
7.500000  0.018187
8.500000  0.016333
9.500000  0.016975
10.500000  0.014621
11.500000  0.014550
12.500000  0.010984
13.500000  0.011839
14.500000  0.009842
15.500000  0.009272
16.500000  0.009914
17.500000  0.010128
18.500000  0.009200
```

2. 数据建图

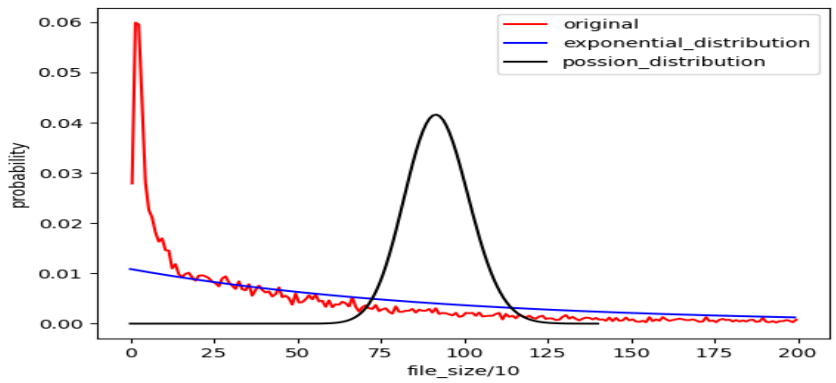
我们在学习 C 语言的时候，没有太学过如何进行这方面的建图，所以我们之中的组员学习一下利用 python 来进行画图。

具体思路：

代码如下

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import factorial

x = []
y = []

with open ("Task1.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        value = [float(s) for s in line.split()]
        x.append(value[0])
        y.append(value[1])

x1 = np.arange(0,200)
y1 = 1/92*np.exp(-x1/92)
##plt.xlim(0, 280000)
##plt.ylim(0, 0.005)
##y_ticks = np.arange(0, 0.005, 0.00001);
##plt.yticks(y_ticks)
plt.xlabel("file_size/10")
plt.ylabel("probability")
plt.plot(x, y, color="red", label="original")
plt.plot(x1, y1, color="blue", label="exponential_distribution")

x2 = np.linspace(0,140,140)
#y2 = [((t)**(i)*np.exp(-1*t))/(np.math.factorial(int(i))) for i in x2]
y2 = (92**x2)*np.exp(-92)/factorial(x2)
plt.plot(x2, y2, color="black",label="possion_distribution")
plt.legend(loc="upper right")
plt.show()
```

根据上面程序，建立如下数据图：



根据单纯的数据图，我们猜测有大概三种分布可能会是这样的分布：指数分布，泊松分布，二项分布。之后我们用参数估计的方法来画出这几种分布的图来进行大概的推测。

(1)指数分布

$f(x) = \frac{1}{\theta} e^{-\frac{x}{\theta}}$ ， x>0

所以利用参数估计的方法来进行参数的计算。

E(X) = θ

之后我们通过 cpp 程序来进行期望的计算。

计算出的具体值为 91.791443（文件大小已经/10）

而具体的结果大致上符合我们画出的图。

(2)泊松分布

$P(X=k) = \frac{\lambda^k}{k!} e^{-\lambda}$ ， k=0,1,2……

泊松分布的期望和方差均为 λ，利用指数分布所计算出均值来进行画图。

根据画出的分布图，我们可以知道原本的图不是泊松分布。

(3)二项分布

根据播送定理，二项分布的分布图大致类似于泊松分布，而且在 n 足够大的时候，两者相同。

3. 总结

根据分析，我们可以知道该问题属于指数分布。

## Task2：分析随着时间的变化，该网站所需的容量的变化趋势

1. 数据处理

根据时间将文件的大小依次求和，每个时间的和就是网站所需的容量。

代码实现如下：

```
while(1){
    if(scanf("%c",&ch)==EOF) break;
    while(ch!='>') scanf("%c",&ch);  scanf("%c",&ch);
    if(ch=='<'){ while(ch!='>') scanf("%c",&ch); scanf("%c",&ch);}
    tot++;
    for(int i=1;i<=3;i++) scanf("%c",&ch),Record[tot].Day.push_back(ch);
    scanf("%c",&ch); scanf("%d ",&Record[tot].day);
    cin>>Record[tot].Mon>>Record[tot].year>>Record[tot].Time;
    Record[tot].mon=M[Record[tot].Mon];
    scanf("%c",&ch);
    while(ch!=')'){
        if(ch>='0'&&ch<='9') Record[tot].size=Record[tot].size*10+ch-'0';
        scanf("%c",&ch);
    }
}
sort(Record+1,Record+tot+1);
```

```
ll sum=0;
int d = Record[1].day;
cerr<<Record[1].mon<<endl;
for(int i=1;i<=tot;i++){
     if (d != Record[i].day)
     {
          cout<<Record[i-1].count(com(Record[1]))<<' ';
          cout << sum <<endl;
          d = Record[i].day;
     }
     sum+=Record[i].size;
}
```

输出的结果部分如下，左边为距离 2013.10.08 的天数，右为截至到当天的所有的论文的大小之和

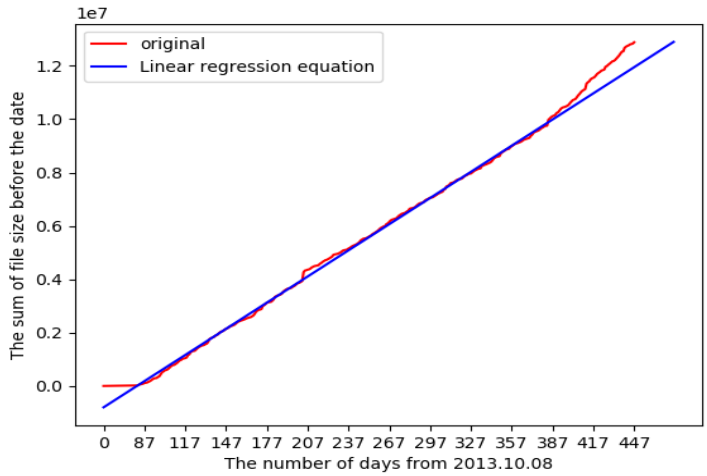
```
0 605
3 1191
13 2074
23 2092
25 2214
30 4424
31 4710
32 5377
36 5750
42 7441
43 8568
46 8719
48 9004
51 9534
60 9606
62 9811
67 13599
70 14292
71 14327
```

2. 数据建图

画图代码具体如下：

```
#with open("Task2.txt", "r")
with open ("Task2_2.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        line = line.strip('\n')
        if index%2 == 0:
            x.append(line)
##              print(line)
##              print(" ")
        else:
            y.append(int(line))
##              print(line)
##              print("\n")
        index = index+1
index = 0;
xx = []
for i in x:
    if index%30 == 0:
        xx.append(i)
    index = index+1
plt.xticks(np.arange(0,len(xx)*30,30), xx)
plt.plot(x, y, color="red", label="original")
x2 = np.arange(0, len(xx)*30)
y2 = 32687*x2-800000.7
plt.plot(x2, y2, color="blue", label="Linear regression equation");
plt.xlabel("The number of days from 2013.10.08")
plt.ylabel("The sum of file size before the date")
plt.legend(loc="upper left")
plt.show()
```

根据以上的程序的输出结果建立数据图。




在建立的图中，我们可以发现图片很近似于线性的曲线。所以我们猜测这

个曲线可能是线性曲线。之后我们利用了最小二乘法来进行线性回归拟合。

$$a = \frac{\Sigma xy - \frac{1}{N}\Sigma x \Sigma y}{\Sigma x^2 - \frac{1}{N}(\Sigma x)^2}$$

$$b = \bar{y} - a\bar{x}$$

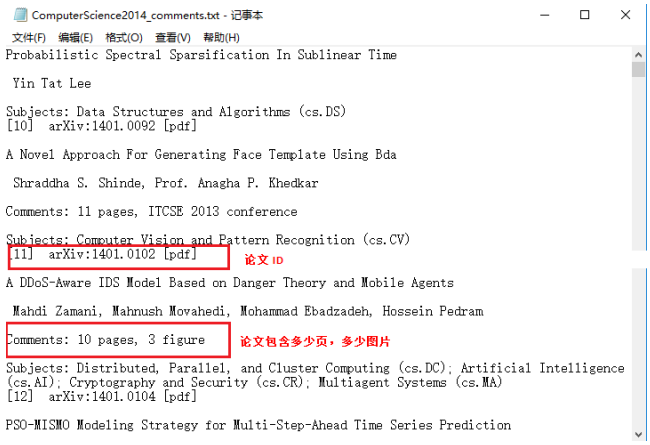
通过计算我们得出了斜率为 32687.022730，代入公式，我们将线性回归的曲线建立，在将两个曲线合并之后，我们可以看出这两个曲线在一定的区间内十分的拟合。

3. 总结

根据图像，我们能够大胆推测该曲线在一定区间内是线性曲线。

## Task3：论文页数的概率分布

1. 数据处理




根据老师给出的文件，我们可以分析知道每个文件的 pages 和 figures 都是在 comments 中。这样我们在设计程序进行数据处理的时候可以只处理每个文件的 comments 的行。

代码实现如下：

```
for (i = 1;i < 100010;i++)
{
    if (i%10 == 0)
    {
        if (sum)
            printf("%d %f\n", i-5, (double)sum/(double)total1);
        sum = 0;
    }
    sum += page[i];
}
```

```
for(i = 0;i < 100010;i++)
{
    if (i%5 == 0)
    {
        if (sum)
            printf("%d %f\n", i-3, (double)sum/(double)total2);
        sum = 0;
    }
    sum += figure[i];
}
```

输出结果部分如下，左为论文页数，右为相应的概率。

```
5 0.327844
15 0.399093
25 0.155268
35 0.068728
45 0.021845
55 0.011427
65 0.004201
75 0.003361
85 0.001680
95 0.001176
105 0.000672
115 0.000672
125 0.000840
135 0.000336
175 0.000336
185 0.000168
195 0.000168
205 0.000168
215 0.000168
```

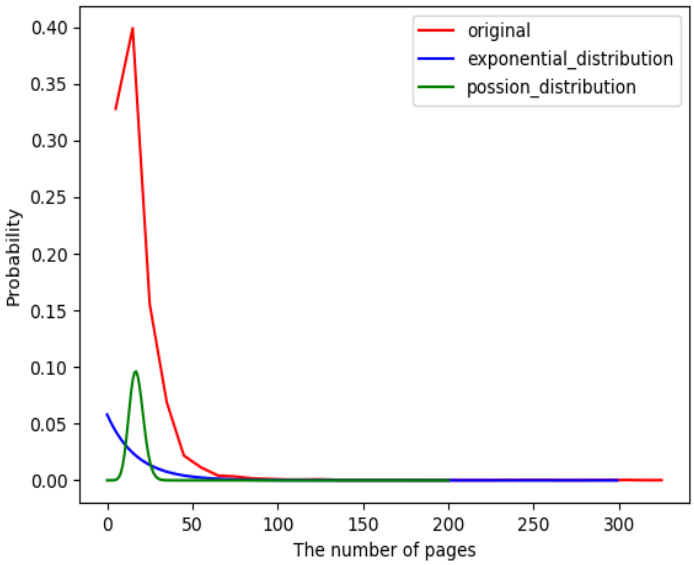
2. 数据建图

通过以上程序得出的结果，我们建立图像。

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import factorial

x = []
y = []

with open ("Task3.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        value = [float(s) for s in line.split()]
        x.append(value[0])
        y.append(value[1])
x2 = np.arange(0, 300)
y2 = 1/17.2196*np.exp(-x2/17.2196)
x3 = np.linspace(0,200, 200)
y3 = (17.2196**x3)*np.exp(-17.2196)/factorial(x3)
plt.plot(x, y, color="red", label="original")
plt.plot(x2, y2, color="blue", label="exponential_distribution")
plt.plot(x3, y3, color="green", label="possion_distribution")
plt.legend(loc="upper right")
plt.xlabel("The number of pages")
plt.ylabel("Probability")
plt.show()
```

同时类似于 task1，我们猜测这个曲线可能为指数分布，泊松分布。

我们通过程序求出 $\lambda = \frac{1}{\theta} = 17.2196$。再来建立图。




其中真正概率分布曲线为红色的曲线，指数分布为蓝色的曲线，泊松分布为绿色曲线。

3. 总结

根据曲线的拟合程度，我们推测该曲线更可能为泊松分布。

## Task4：以及论文图形数目的概率分布

1. 数据处理

该 task 的数据处理在 task3 中已经同时求出。

输出结果如下，左为论文图形数目，右为相应概率。

```
2  0.396962
7  0.400882
12  0.135261
17  0.038961
22  0.014947
27  0.005636
32  0.002695
37  0.001715
47  0.000980
52  0.000490
57  0.000490
67  0.000245
72  0.000245
87  0.000245
```

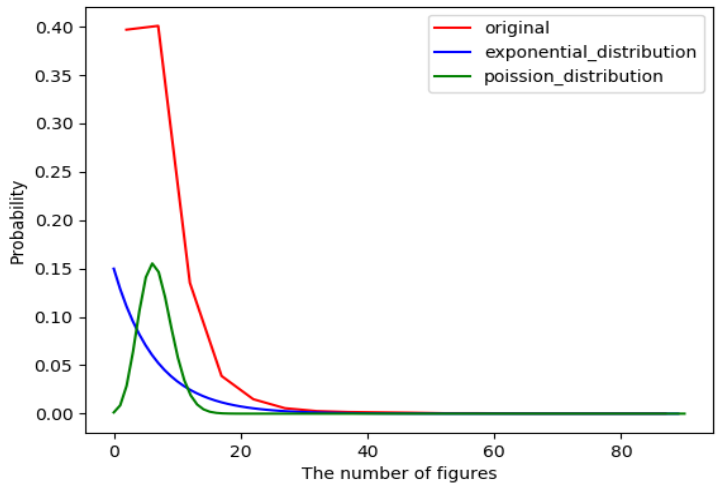
2. 数据建图

通过以上程序得出的结果，我们建立图像。

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import factorial

x = []
y = []

with open ("Task4.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        value = [float(s) for s in line.split()]
        x.append(value[0])
        y.append(value[1])
x2 = np.arange(0, 90)
y2 = 1/6.67111*np.exp(-x2/6.67111)
x3 = np.linspace(0,90,90)
y3 = (6.67111**x3)*np.exp(-6.67111)/factorial(x3)
plt.plot(x, y, color="red", label="original")
plt.plot(x2, y2, color="blue", label="exponential_distribution")
plt.plot(x3, y3, color="green", label="poission_distribution")
plt.legend(loc="upper right")
plt.xlabel("The number of figures")
plt.ylabel("Probability")
plt.show()
```

同时类似于 task1，我们猜测这个曲线可能为指数分布，泊松分布。

我们通过程序求出 $\lambda = \frac{1}{\theta} = 6.67111$。再来建立图。




其中，其中真正概率分布曲线为红色的曲线，指数分布为蓝色的曲线，泊松分布为绿色曲线。

3. 总结

根据曲线的拟合程度，我们推测该曲线更可能为泊松分布。

## 课题总结

除了 Task2，猜测的曲线与实际曲线拟合度高外，其他 3 个 Tasks，拟合度一点也不高，甚至可以说有点牵强。可以看到，Task1、3、4 的猜测出来的曲线在“尾巴”部分才与真实的曲线相拟合，而“头部”部分相差甚远，我们讨论后，觉得原因应该是从整体的数据猜测出来的曲线，它主要与大部分数据拟合，而与小部分数据的误差就相对较大，所以造成了这样的结果。我们其实也尝试过忽略

尾部的概率很小的数据，但效果还是不尽人意，所以还是保留最开始的完整数据得出的猜测曲线。其实，我们觉得或许 Task1、2、3 真正的概率分布其实并不是什么指数分布或者泊松分布，它就是具有自己个性的分布。

参考文献：

[1] 盛骤 谢式千 潘承毅. 《概率论与数理统计》. 高等教育出版社. 2008