Python 全栈 400 之统计学练习

381 什么是一个事件？

抛硬币出现正面就可以定义为一个事件 X；抛硬币出现反面也能被定义为一个事件 Y；抛硬币 10 次，正面朝上次数也可定义为一个事件 Z；在101班测量了40名学生的身高，学生的平均身高也可以被定义为一个事件 H.

382 什么是概率？

概率描述某个事件发生的概率，比如上面事件 X 发生的概率为 0.4，则对应的事件 Y 发生的概率就为 0.6，事件 X 和 事件 Y 为一对对立事件。抛硬币 10 次，正面朝上次数为事件 Z ，可能的取值为 0 到 10 ，发生 0 次的概率一般记做 P(X=0) ，它等于 `0.6^10`；出现 1 次的概率 P(X=1) ，等于 `0.4 * 0.6 ^ 9`，后面依次类推。

383 什么是离散型随机变量？

比如上面的事件 Z ，可能取值为 0 到 10， 这是一个**离散型随机变量**，因为事件 Z 可能的取值个数是有限的。

384 什么是连续型随机变量？

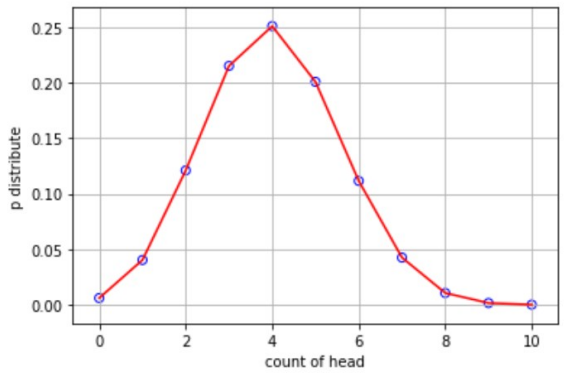
101班 40 名学生的平均身高对应的事件 H 取值就有各种可能性，因此事件 H 取值为**连续型随机变量。**

385 概率分布描述的是什么？以二项分布为例阐述

概率分布表示某个事件的所有可能取值对应的概率情况。求出每个可能的取值对应的概率，就相当于得到事件 Z 的概率分布。此处事件 Z 的概率分布实际上就是我们常见的 **二项分布**，下面绘制出事件 Z 在正面发生概率为 0.4 情况下的概率分布图：

复制

```
ph = 0.4
pt = 1 - ph
sumn = 10

zi = range(0,sumn+1)

def combine_number(sumn, headn):
    return math.factorial(sumn) / math.factorial(headn) / math.factorial(
sumn - headn)

pi = [combine_number(10,headn)* math.pow(ph,headn) * math.pow(pt,sumn - h
eadn) for headn in zi]

plt.plot(zi,pi,color='r')
plt.scatter(zi,pi,color='', marker='o',edgecolor='b')
plt.grid()
plt.xlabel('count of head')
plt.ylabel('p distribute')
plt.show()
```



看到正面出现 4 次的概率最大为 0.25.

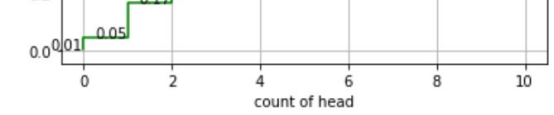
386 累积分布函数 x, y 的意义是什么？

如上事件 Z 出现次数不大于 4 次的概率累加和，记做 F(4)，等于 P(Z<=4) 的概率，逐个计算出 F(n)，n 等于 0 到 10， 这样就得到一个累积分布函数，离散型变量的累积分布函数被称作概率质量函数。

代码接着第 6 节，下面绘制出事件 Z 的概率质量函数：

复制

```
cfd = np.cumsum(pi)
plt.step(zi,cfd,color='g')
for i,j in zip(zi,cfd):
    plt.text(x=i-0.75,y=j,s=round(j,2))

plt.xlabel('count of head')
plt.ylabel('cumulation p')
plt.grid()
plt.show()
```



387 NumPy 求平均值、中位数、标准差、方差

描述性变量是对已有数据的基本情况统计量，比如测量出 101 班 40 名学生的身高，我们就能基于这些数据，计算出以下描述性统计量：

- 平均值
- 中位数
- 标准差
- 方差

假定下面为 101 班级 40 名学生测量身高值：

复制

```
hs = array([189., 158., 178., 169., 146., 169., 168., 169., 160., 177., 1
61.,
       191., 155., 177., 171., 183., 168., 181., 189., 180., 170., 168.,
       170., 169., 176., 163., 136., 165., 179., 150., 186., 160., 177.,
       154., 165., 186., 168., 182., 167., 162.])
```

使用 NumPy 很容易计算出 40 名学生的平均身高值：169.8

复制

```
np.mean(hs)
```

中位数：169.0，按照身高从低到高排序后，位于中间的数值。

复制

```
np.median(hs)
```

它与均值不同，某些场景选用中位数会更加合适，尤其数据分布不均匀时。比如中等收入家庭占多，富豪还是少数但是财富值远多于中等收入家庭。如果取所有样本的均值就会高于大多数中等家庭收入均值，此时选用样本中位数更能贴近家庭实际收入情况。

方差：144.16

复制

```
np.var(hs)
```

标准差：12.0

复制

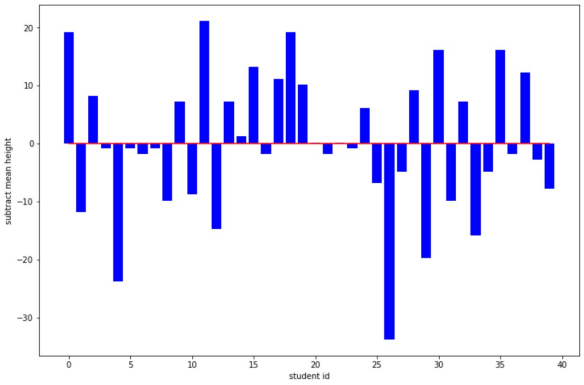
```
np.std(hs)
```

方差取根号得到标准差，它们表示数据偏离平均值的程度，标准差越小表示数据整体更靠近均值，反之越偏离均值，也就更加分散。

展示 40 位学生身高数据偏离平均值的柱状图：

复制

```
import matplotlib.pyplot as plt
import numpy as np
import math

plt.figure(figsize=(12,8))
plt.bar(np.arange(40),hs-np.mean(hs),color='blue')
plt.plot(np.arange(40),np.repeat(0,40),color='r')
plt.xlabel('student id')
plt.ylabel('subtract mean height')
```



从上图看出偏离平均值（红线水平线）最大的样本 id 位于 25 到 30 间，偏离值为 30 多。

388 分位数和箱型图传递什么信息？

快速了解数据的分布最便捷的方法之一就是通过分位数，绘制箱型图。

常用的分位数有上、下分位数和中位数，分别对应数据 75%，25%，50% 比例处。

使用 NumPy 分别求出 40 位学生身高的上、下分位数和中位数，依次为：178.25，169.0，169.0.

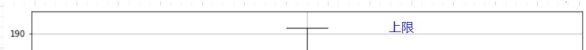
复制

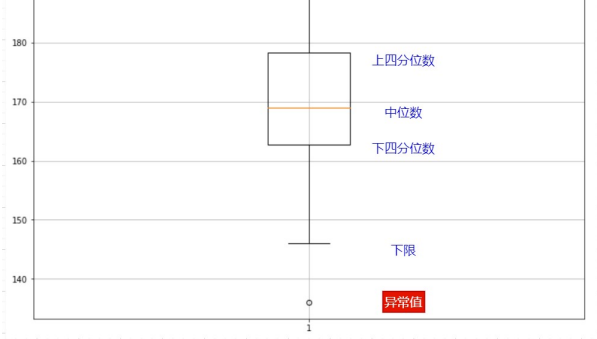
```
p3 = np.percentile(hs,75)
p1 = np.percentile(hs,25)
p2 = np.percentile(hs,50)
```

接下来绘制箱型图，进一步获取到更多数据分布的信息：

复制

```
plt.figure(figsize=(12,8))
plt.grid()
d = plt.boxplot(hs)
```





从箱型图中看出如上标注的信息，位于上、下限之外的样本点是异常值，用圆圈表示，如上图所示。

389 期望和频率分布直方图的案例

101 班 40 名学生的身高值是连续性随机变量，很显然研究具体某个身高值的概率没有意义。一般的，将样本集合的最小值、最大值对应的区间均分为若干等分，求出某个样本落在对应区间的概率值，记做此身高值的概率。

使用 Matplotlib 绘制频率分布直方图，得到 10 个区间，以及落入每个区间的样本个数：

复制

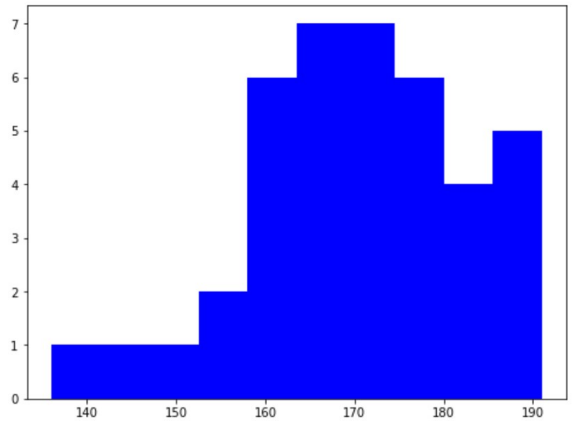
```
plt.figure(figsize=(8,6))
hsh = plt.hist(hs,color='b')
hsh
```

注意 hist 方法返回的结果为一个元组，第一个元素为区间内样本的个数，第二个元素表示划分的10个区间，例如落在区间[136.,141.5] 的样本个数为 1， 落在区间 [152.5, 158] 内元素个数为 2.

复制

```
(array([1., 1., 1., 2., 6., 7., 7., 6., 4., 5.]),
 array([136. , 141.5, 147. , 152.5, 158. , 163.5, 169. , 174.5, 180. ,
        185.5, 191. ]),
 <a list of 10 Patch objects>)
```

频率分布直方图就是绘制区间与落在区间内样本个数的直方图，如下所示：



基于频率分布直方图理解期望更容易，从划分的10个子区间内分别选取一个代表性数值，然后求出每个子区间的概率，取两者的乘积求和便是身高的期望值，计算过程如下：

复制

```
hsp  = hsh[0] / sum(hsh[0]) # 计算每个子区间的概率
hsv = [ (hsh[1][i] + hsh[1][i+1])/2.  for i in range(len(hsh[1])-1)] #选取子区间的中位数作为此区间的代表值
ex = sum(hsp * np.array(hsv))
ex
```

得到期望值为：170.2375，也就是基于 40 个学生身高样本，我们对学生身高的期望值大约为 170.

390 如何求协方差的案例

上面介绍的都是对单变量的统计分析，而协方差和相关系数分析的是两个随机变量间的相关程度，是对两个随机变量的联合分布的概率度量，给出两个随机变量怎样一起变化。

如下为 101 班 40 名学生某次期末考试的数学和物理分数：

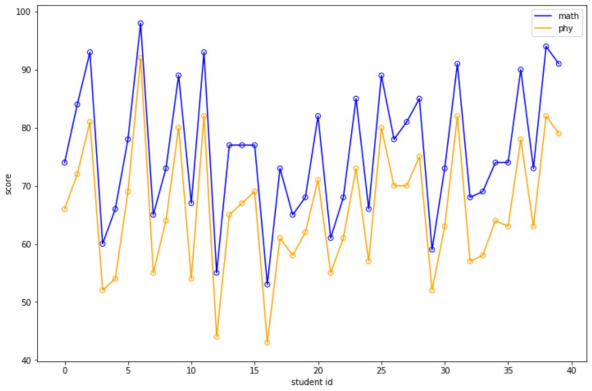
复制

```
(array([74, 84, 93, 60, 66, 78, 98, 65, 73, 89, 67, 93, 55, 77, 77, 77, 53
,
        73, 65, 68, 82, 61, 68, 85, 66, 89, 78, 81, 85, 59, 73, 91, 68, 69
,
        74, 74, 90, 73, 94, 91]),
 array([66, 72, 81, 52, 54, 69, 92, 55, 64, 80, 54, 82, 44, 65, 67, 69, 43
,
        61, 58, 62, 71, 55, 61, 73, 57, 80, 70, 70, 75, 52, 63, 82, 57, 58
,
        64, 63, 78, 63, 82, 79]))
```

图形展示学生的数学成绩和物理成绩：

复制

```
plt.figure(figsize=(12,8))
plt.plot(maths,color='b',label='math')
plt.scatter(np.arange(40),maths, color='', marker='o',edgecolor='b')
plt.plot(phys,color='orange',label='phy')
plt.scatter(np.arange(40),phys, color='', marker='o',edgecolor='orange')
plt.xlabel('student id')
plt.ylabel('score')
plt.legend()
```



求两个变量的协方差：

复制
```
np.cov(maths,phys)
```

结果如下，分别表示自身与自身，自身与对方的相关程度，不过估计看完没啥感觉，我们再来一组对比分析。

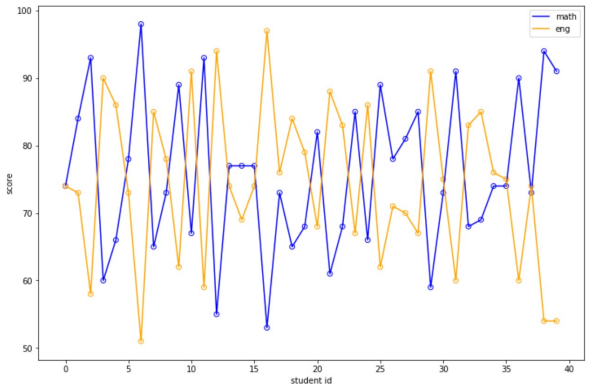
复制
```
array([[153.01609195, 156.49195402],
       [156.49195402, 164.59885057]])
```

再拿 40 位学生的英语成绩与数学成绩做对比：

复制
```
array([74, 73, 58, 90, 86, 73, 51, 85, 78, 62, 91, 59, 94, 74, 69, 74, 97
,
       76, 84, 79, 68, 88, 83, 67, 86, 62, 71, 70, 67, 91, 75, 60, 83, 85
,
       76, 75, 60, 74, 54, 54])
```

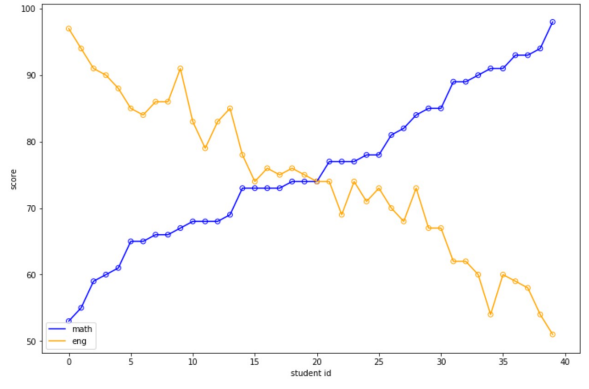
绘制图形：

复制
```
plt.figure(figsize=(12,8))
plt.plot(maths,color='b',label='math')
plt.scatter(np.arange(40),maths, color='', marker='o',edgecolor='b')
plt.plot(engs,color='orange',label='eng')
plt.scatter(np.arange(40),engs, color='', marker='o',edgecolor='orange')
plt.xlabel('student id')
plt.ylabel('score')
plt.legend()
```



有些杂乱，按照数学成绩升序排序后，重新绘制图形：

复制
```
### 按照分数升序排序后;
maths_ = np.sort(maths)
engs_arg = np.argsort(maths)
engs_ = engs[engs_arg]
plt.figure(figsize=(12,8))
plt.plot(maths_,color='b',label='math')
plt.scatter(np.arange(40),maths_, color='', marker='o',edgecolor='b')
plt.plot(engs_,color='orange',label='eng')
plt.scatter(np.arange(40),engs_, color='', marker='o',edgecolor='orange')
plt.xlabel('student id')
plt.ylabel('score')
plt.legend()
```



再求出数学和英语的协方差：

复制
```
np.cov(maths,engs)
```

结果为：

```
array([[ 133.83589744, -134.06153846],
       [-134.06153846,  140.2974359 ]])
```

两个协方差对比，可以看出学生的数学成绩和物理成绩正相关，数学成绩和英语成绩负相关。

391 协方差和相关系数的关系

协方差除以两个变量的方差乘积，得到相关系数。

```
np.corrcoef(maths,phys)
```

数学和物理高度正相关，也就是说数学分数高，物理分数就高。

```
array([[1.        , 0.98582444],
       [0.98582444, 1.        ]])
```

```
np.corrcoef(maths,engs)
```

数学和英语高度负相关，数学分数高，英语分数一般不高。

```
array([[ 1.        , -0.97834724],
       [-0.97834724,  1.        ]])
```

相关系数的取值范围为 [-1,1] ，大于 0 表示正相关，小于 0 表示负相关，等于 0 表示两个事件相关独立，无任何关系。

392 使用 scipy 做 t 检验的案例

假定本次期末考试高三年级的数学和物理成绩都服从正态分布，且**方差相等**。如上已经知道 101 班数学考试的平均成绩和标准差分别为 `X = 75.9` ,`Sx = 11.4` ，40 名学生的物理平均成绩为 `Y = 66.1`, Sy =11.1 `.

试问本次期末考试全校的数学和物理成绩的平均分有无显著差异？

这是一个典型的双样本，满足正态分布且方差相等的假设检验，判断两个正态分布的期望是否相同，这就是一个 t 检验问题。

使用 `Scipy` 求解双样本同方差的 t 检验问题，原假设 H0：数学和物理成绩的平均分无显著差异，H1：有显著差异。

```
import numpy as np
from scipy import stats

mean1 = 75.9
mean2 = 66.1

std1 = 11.4
std2 = 11.1

nobs1 = 40
nobs2 = 40

modified_std1 = np.sqrt(np.float32(nobs1)/np.float32(nobs1-1)) * std1
modified_std2 = np.sqrt(np.float32(nobs2)/np.float32(nobs2-1)) * std2

(statistic, pvalue) = stats.ttest_ind_from_stats(mean1=mean1, std1=modifi
ed_std1, nobs1=nobs1, mean2=mean2, std2=modified_std2, nobs2=nobs2)
(statistic, pvalue)
```

得到结果为：

```
(3.8463804246241637, 0.00024347752844308177)
```

假设我们显著性水平 α=0.05，pvalue 显著的小于 0.05，所以我们拒绝原假设 H0，也就是认为数学和物理成绩的期望存在显著性差异。

解释下 p 值，p 值表示原假设发生概率大小。p 值越小说明原假设情况发生的概率就越小。

393 完成 F 检验的案例

t 检验是为检验均值是否有显著性差异，F 检验是为检验方差是否有显著性差异。

如下为两种车型近10天的销量情况，检验两种车型的销量方差是否存在显著差异，原假设 H0 为两种车型的销量方差无显著差异， H1 为方差有显著差异。

```
car_a = array([28, 72, 73, 62, 27, 91, 76, 63, 95, 20])
car_b = array([86, 40, 60, 37, 64, 51, 39, 53, 26, 81])
```

使用 F 检验过程如下：

```
def varf(a):
    return sum(np.power(a - np.mean(a),2)) / (len(a)-1)

f = varf(car_a) / varf(car_b)
f
```

统计值 f 为 ： 1.91

参考 F 检验表
（ https://wenku.baidu.com/view/fafb62536fdb6f1aff00bed5b9f3f90f77c64d5b.html ），在显著性上水平等于 0.05 ，样本个数都为 9 的情况下关键值为 3.18，统计值 f 小于 3.18 ，所以接收原假设 H0，即认为两种车型的方差无显著性差异。

394 如何做卡方检验的案例

卡方检验用于检测分类型变量是否符合期望频数。例如，投掷一个 6 面筛子共 60 次，期望每一

面出现的频次都为 10. 但是不确定这个筛子是否动过手脚，于是投掷 60 次，得到实验数据如下：

```
array([11,  8,  9,  8, 10, 14])
```

使用卡方检验判断原假设 H0: 未动过手脚，H1: 动过手脚

```
import scipy.stats as ss
obs= np.array([11,  8,  9,  8, 10, 14])
exp=np.repeat(10,6)
#拒绝域 5% 的显著水平,自由度 5
jjy=ss.chi2.isf(0.05,5)

#卡方
kf=ss.chisquare(obs,f_exp=exp).statistic

jjy, kf
```

结果：

```
jjy, kf = (11.1, 2.6)
```

5% 的显著水平，自由度 5 的拒绝域为：大于 11.1，卡方检验计算的卡方值为 2.6 ，不在拒绝域内，所以接收原假设，即筛子未动过手脚。

395 什么是独立同分布？

独立同分布这个概念在各大书籍、网络中非常常见，它的英文名称：independent and identically distributed, 简称为i.i.d. 首先确保以后读文献、看博客等要认识这种简写。

分别解释什么是独立、什么是同分布。

抛掷一枚硬币，记出现正面为事件 X ，事件 X 发生的概率为 $P(X)$ 且为 0.4 . 接下来开始做抛掷硬币实验，第一次抛掷硬币出现正面的概率为 0.4，第二次抛掷硬币出现正面的概率 $ P(X2) 也为0.4，第i次正面出现概率 P(Xi)也为0.4，也就说每第i次抛掷与前面i-1$ 次抛掷无任何关系，与它们相互独立。

如果抛掷一枚智能硬币，如果第 $i-1$ 次出现反面，则第 i 次一定为正面；如果第 $i-1$ 次出现正面，则第 i 次一定为反面。那么这就不是独立的，第 i 次出现正反面强依赖于第 $i-1$ 次的实验结果。

同分布是指每次抛掷试验，使用的都是这枚正面出现概率 $P(X)$ 为 0.4 的硬币。相对的，如果第 $i-1$ 次使用这枚硬币，而第 i 次抛掷却实用正面出现概率 $P(X)$ 为 0.6 的硬币，那么这就不是同分布。

396 为什么要关心分布？

大部分机器学习算法都是根据已有历史数据，去学习它们的分布规律，也就是分布的参数。一旦学习到分布的参数后，当再进来新的、未知的数据时，学到的算法模型便会预测或决策出一个结果。这是大部分机器学习的学习过程。

请留意，评价机器学习模型好快使用的数据是新进来的、对模型完全未知的数据。

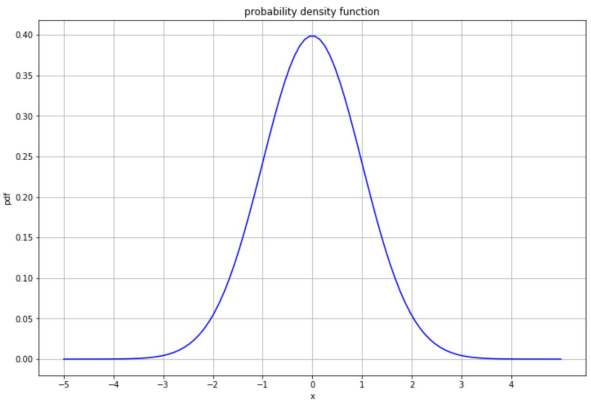
考虑下面这种情况，如果我们再拿训练使用的数据来评价模型好快时，得分肯定高，但是完全没有意义，相信也不会有人这么做，因为它们已经对模型完全学习到、完全已熟悉。

再考虑另一种情况，如果测试用的数据来自完全不同的数据分布，模型预测它们的结果得分往往不会好，虽然也会得到一个分数。测试数据集的分布和训练数据集的数据分布差异太大，训练的模型即便泛化的再好，预测与己分布差异很大数据时也无能为力。

基于以上两种极端情况，我们的希望便是测试数据集要尽可能匹配训练模型所使用的的数据分布，在这个前提下，再去优化调参模型才更有意义，努力才不会白费。

397 正态分布和概率密度图等

提到数据分布，大多数读者可能脑海里会闪现出下面这幅图，不过长时间没接触过数理统计的读者可能会遗忘 y 轴含义。



假定 x 是一个连续型随机变量，y 就表示 x 的概率密度函数，说的直白些，它是取值密集情况的一种度量。下图是标准正态分布的 pdf 曲线，在 x 等于 0 处，y 值最大。

首先附上绘制上面图形的过程，导入包：

```
import matplotlib.pyplot as plt
from scipy import stats
```

使用 `scipy.stats` 的 `norm` 方法，`loc`和`scale` 默认值分别 0 和 1，也就是标准正态分布情况。

```
```

```
plt.figure(figsize=(12,8))
x = np.linspace(-5,5,100)
y = stats.norm.pdf(x) # pdf 表示概率密度函数
plt.grid()
plt.xlabel('x')
plt.ylabel('pdf')
plt.title('probability density function')
plt.xticks(ticks=np.arange(-5,5))
plt.plot(x,y,color='blue')
```

使用 `scipy` 会很方便地求出各个关键取值，比如计算 x 等于 0.5 时概率密度值，结果为：0.352

复制

```
stats.norm.ppf(0.5)
```

使用 `ppf` 方法，求出累计分布函数（也就是曲线与x轴所围成面积）等于 0.5 时，x 的值，显然面积值等于一半时，x 等于 0.5.

复制

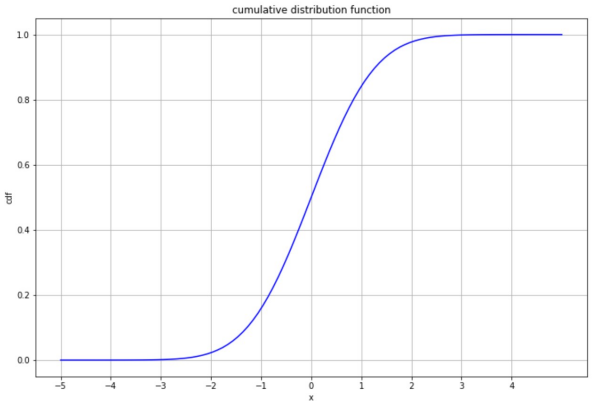
```
stats.norm.ppf(0.5)
```

求出标准正态分布的均值和标准差：

复制

```
stats.norm.mean() # 0.
stats.norm.std() # 1.
```

上面提到累计分布函数，它的定义为 $P(X < x)$ ，密度的累积，绘制标准正态分布的累计分布函数：



绘制代码如下：

复制

```
plt.figure(figsize=(12,8))
x = np.linspace(-5,5,100)
y_cdf = stats.norm.cdf(x)
plt.grid()
plt.xlabel('x')
plt.ylabel('cdf')
plt.title('cumulative distribution function')
plt.xticks(ticks=np.arange(-5,5))
plt.plot(x,y_cdf,color='blue')
```

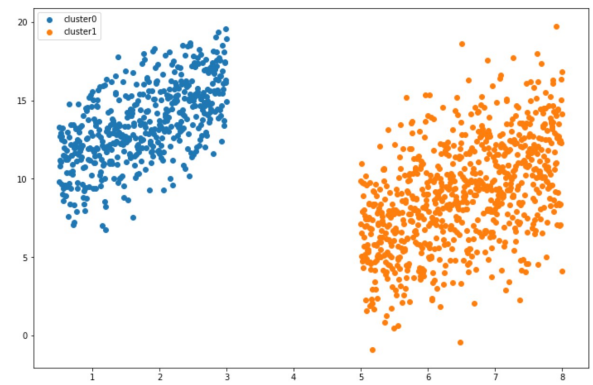
正态分布应用广泛，现实中当我们对数据分布一无所知时，往往假定数据符合正态分布。今天介绍一个使用正态分布生成实验使用的几簇数据集，用于做聚类等任务的实验数据。

基本原理：想生成几簇数据，就创建几条线段，然后在 `y` 上做一个高斯随机偏移。

生成两簇数据如下，使用的另一个关键方法 `rvs`，表示生成满足指定分布、参数指定个数的样本。

复制

```
plt.figure(figsize=(12,8))
blob1 = 500
x1 = np.linspace(0.5,3,blob1)
y1 = 2 * x1 + 10 + stats.norm.rvs(0.,2.,size=(blob1,)) # rvs 生成正态分布的
blob1 个数据样本

blob2 = 800
x2 = np.linspace(5,8,blob2)
y2 = 2 * x2 - 4 + stats.norm.rvs(0.,3.0,size=(blob2,))

plt.scatter(x1,y1,label='cluster0')
plt.scatter(x2,y2,label='cluster1')
plt.legend()
plt.show()
```

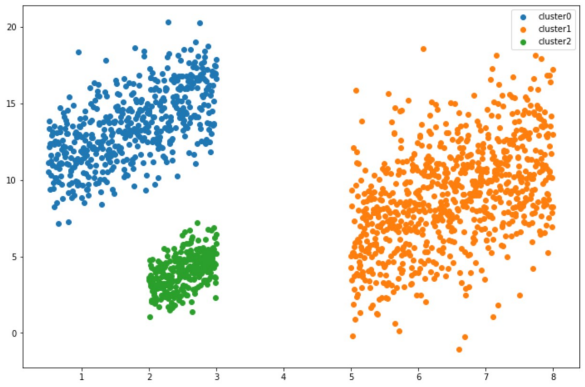


同样方法，生成如下三簇数据：

复制

```
plt.figure(figsize=(12,8))
blob1 = 500
x1 = np.linspace(0.5,3,blob1)
```

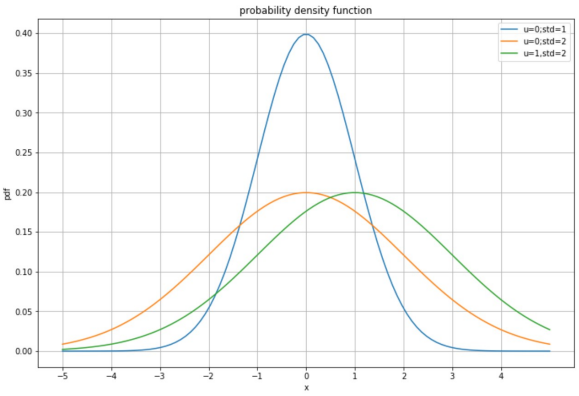
```
y1 = 2 * x1 + 10 + stats.norm.rvs(0.,2.,size=(blob1,))

blob2 = 800
x2 = np.linspace(5,8,blob2)
y2 = 2 * x2 - 4 + stats.norm.rvs(0.,3.0,size=(blob2,))

blob3 =300
x3 = np.linspace(2,3,blob3)
y3 = 2 * x3 -1 + stats.norm.rvs(0.,1.0,size=(blob3,))

plt.scatter(x1,y1,label='cluster0')
plt.scatter(x2,y2,label='cluster1')
plt.scatter(x3,y3,label='cluster2')

plt.legend()
plt.show()
```



正态分布的两个参数 `loc` 和 `scale` 分别表示均值和标准差，几种不同组合的对比分析：

复制

```
plt.figure(figsize=(12,8))
x = np.linspace(-5,5,100)
y1 = stats.norm.pdf(x)
y2 = stats.norm.pdf(x,loc=0.0,scale=2.)
y3 = stats.norm.pdf(x,loc=1.0,scale=2.)
plt.grid()
plt.xlabel('x')
plt.ylabel('pdf')
plt.title('probability density function')
plt.xticks(ticks=np.arange(-5,5))
plt.plot(x,y1,label='u=0;std=1')
plt.plot(x,y2,label='u=0;std=2')
plt.plot(x,y3,label='u=1,std=2')
plt.legend()
plt.show()
```



方差越大，取值越离散，表现出来的形状就更矮胖。

398 拉普拉斯分布和正态分布的对比分析

标准的正态分布概率密度函数为：

$$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2\pi}}$$

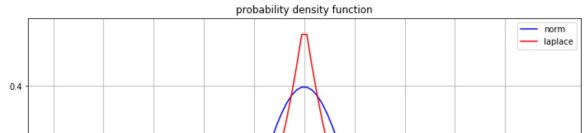
标准的拉普拉斯分布的概率密度函数为：

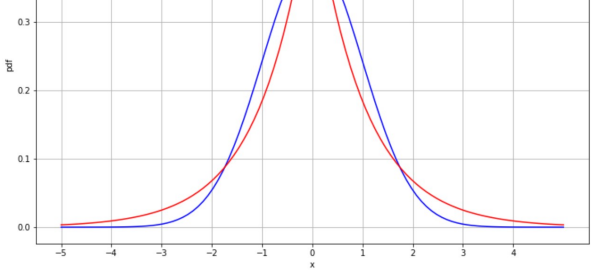
$$f(x) = \frac{1}{2}\exp(-|x|)$$

如果仅仅对比上面的概率密度公式仍然没有感觉，下面绘图对比两者的区别：

复制

```
plt.figure(figsize=(12,8))
x = np.linspace(-5,5,100)
y1 = stats.norm.pdf(x)
y2 = stats.laplace.pdf(x)
plt.grid()
plt.xlabel('x')
plt.ylabel('pdf')
plt.title('probability density function')
plt.xticks(ticks=np.arange(-5,5))
plt.plot(x,y1,color='blue',label='norm')
plt.plot(x,y2,color='red',label='laplace')
plt.legend()
plt.show()
```

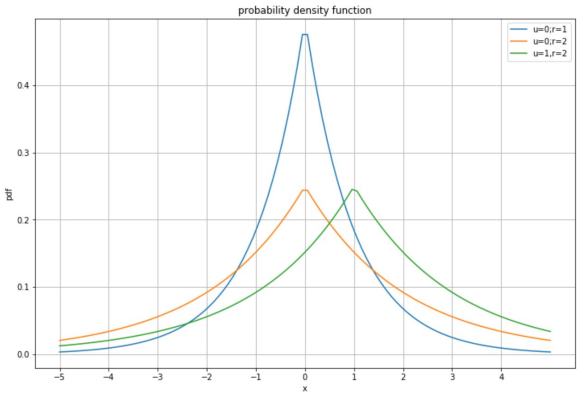




比较顶端，正态分布相比拉普拉斯分布更加平滑，拉普拉斯概率分布却形成一个尖端。

关于两者的对比应用之一就是正则化，其中 `L1` 正则化可看做是拉普拉斯先验，`L2` 正则化看作是正态分布的先验。所以要想深度理解正则化，首先要理解这两个概率分布。

拉普拉斯的两个形状参数与正态分布意义相似：



绘制上图的代码：

复制

```
plt.figure(figsize=(12,8))
x = np.linspace(-5,5,100)
y1 = stats.laplace.pdf(x)
y2 = stats.laplace.pdf(x,loc=0.0,scale=2.)
y3 = stats.laplace.pdf(x,loc=1.0,scale=2.)
plt.grid()
plt.xlabel('x')
plt.ylabel('pdf')
plt.title('probability density function')
plt.xticks(ticks=np.arange(-5,5))
plt.plot(x,y1,label='u=0;r=1')
plt.plot(x,y2,label='u=0;r=2')
plt.plot(x,y3,label='u=1,r=2')
plt.legend()
plt.show()
```

399 伯努利分布

理解完相对复杂一些的两种分布，接下来伯努利相对更加简单，首先它描述的是离散型变量且发生1次的概率分布，且 X 取值只有 2 个，要么为 0，要么为 1. 且 $P(X=1)=p$；$P(X=0)=1-p$，其中 p 为此分布的唯一参数：

$$f(k)=\begin{cases}1-p & \text{if } k=0\\ p & \text{if } k=1\end{cases}$$

复制

```
bern = stats.bernoulli(0.4)
bern.rvs(size=(10,))
```

创建分布参数 $p=0.4$ 的伯努利分布，生成满足此分布的 10 个样本点：

复制

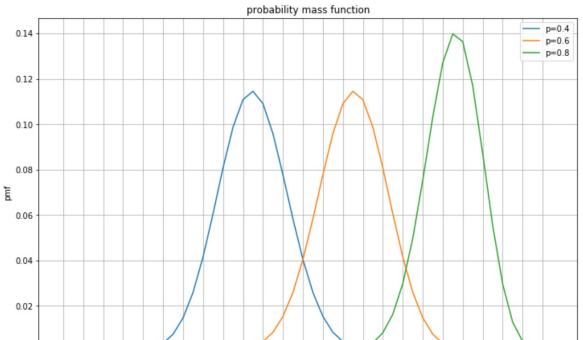
```
array([0, 0, 1, 0, 1, 1, 0, 0, 1, 0])
```

400 二项分布和概率密度图

二项分布也是假设试验只有两种结果，分布参数 p ，且 $P(X=1)=p$；$P(X=0)=1-p$. 不过请注意，二项分布描述的事件是试验独立重复地进行 n 次，且出现 $X=1$ 的次数为 x 次的概率：

$$f(k)=\binom{n}{k}p^k(1-p)^{n-k}$$

独立的重复 50 次试验，三个不同 p 参数的对比分析图如下所示：



绘图代码如下：

复制

```
plt.figure(figsize=(12,8))
x = np.arange(1,51)
y1 = stats.binom.pmf(x,p=0.4,n=50)
y2 = stats.binom.pmf(x,p=0.6,n=50)
y3 = stats.binom.pmf(x,p=0.8,n=50)
plt.grid()
plt.xlabel('x')
plt.ylabel('pdf')
plt.title('probability mass function')
plt.xticks(ticks=np.arange(1,51,2))
plt.plot(x,y1,label='p=0.4')
plt.plot(x,y2,label='p=0.6')
plt.plot(x,y3,label='p=0.8')
plt.legend()
plt.show()
```

三种不同分布参数的二项分布均值和标准差分别为：

复制

```
mean,std = stats.binom.stats(n=50,p=0.4)
mean,std
#(array(20.), array(12.))
```

401 均匀分布

均匀分布的变量可以是离散的，也可以是连续的。

离散型变量均匀分布的概率质量函数为：

$$f(k) = \frac{1}{high - low}$$

复制

```
uniform_int = stats.randint(1,10)
uniform_int.mean(),uniform_int.std() # 均值，标准差
# (5.0, 2.581988897471611)
```

402 泊松分布

假设已知事件在单位时间（或者单位面积）内发生的平均次数为 λ，则泊松分布描述了事件在单位时间（或者单位面积）内发生的具体次数为 k 的概率。

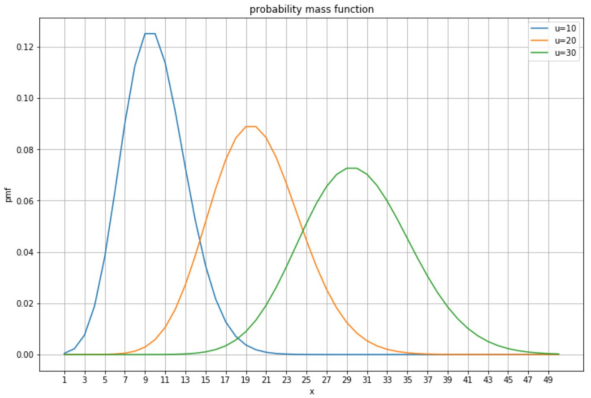
概率质量函数如下，其中 $k > 0$：

$$f(k) = \exp(-\mu)\frac{\mu^k}{k!}$$

分布参数 μ 表示发生次数的期望值，如下绘制不同μ 值下的概率质量分布图：

复制

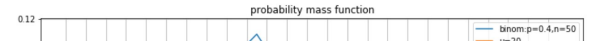
```
plt.figure(figsize=(12,8))
x = np.arange(1,51)
y1 = stats.poisson.pmf(x,10.)
y2 = stats.poisson.pmf(x,20)
y3 = stats.poisson.pmf(x,30)
plt.grid()
plt.xlabel('x')
plt.ylabel('pmf')
plt.title('probability mass function')
plt.xticks(ticks=np.arange(1,51,2))
plt.plot(x,y1,label='u=10')
plt.plot(x,y2,label='u=20')
plt.plot(x,y3,label='u=30')
plt.legend()
plt.show()
```

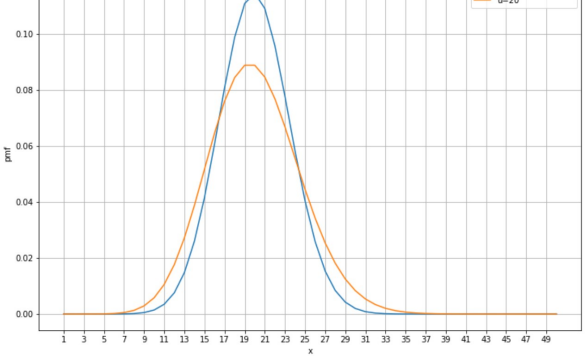


泊松分布与二项分布的对比如下：

复制

```
plt.figure(figsize=(12,8))
x = np.arange(1,51)
y1 = stats.binom.pmf(x,p=0.4,n=50)
y2 = stats.poisson.pmf(x,20)
plt.grid()
plt.xlabel('x')
plt.ylabel('pmf')
plt.title('probability mass function')
plt.xticks(ticks=np.arange(1,51,2))
plt.plot(x,y1,label='binom:p=0.4,n=50')
plt.plot(x,y2,label='u=20')
plt.legend()
plt.show()
```





泊松分布在现实生活中的应用也极为广泛，比如交通流的预测，医学中某个区域内某种细胞的个数等等，感兴趣的读者可去：https://www.zhihu.com/question/26441147 查看更加详细的实际应用解释，在此不再详细展开。

403 指数分布

若事件 X 服从泊松分布，则该事件前后两次发生的时间间隔事件 T 服从指数分布。由于时间间隔是浮点数，因此指数分布是连续分布，其概率密度函数为：

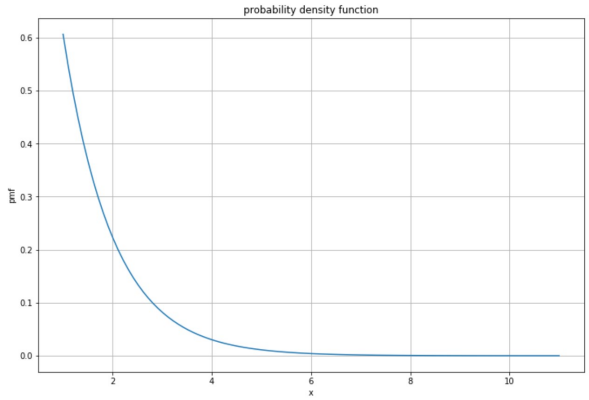
$$p(t;\lambda)=\begin{cases}0, & t<0\\ \dfrac{\lambda}{\exp(\lambda t)}, & t\geq 0\end{cases}$$

标准形式为如下：

$$f(x)=\exp(-x)$$

复制

```
plt.figure(figsize=(12,8))
x = np.linspace(1,11,100)
y1 = stats.expon.pdf(x,0.5)
plt.grid()
plt.xlabel('x')
plt.ylabel('pmf')
plt.title('probability density function')
#plt.scatter(x,y1,color='',edgecolor='orange')
plt.plot(x,y1)
plt.show()
```



指数分布的期望和方差分别为：$\frac{1}{\lambda}$ 和 $\frac{1}{\lambda^2}$

下一章

互动评论

说点什么

评论

The Scrapper 1 个月前

讲的也挺好的~

鼓掌