# [概率论01 计数](http://www.cnblogs.com/vamei/p/3180875.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

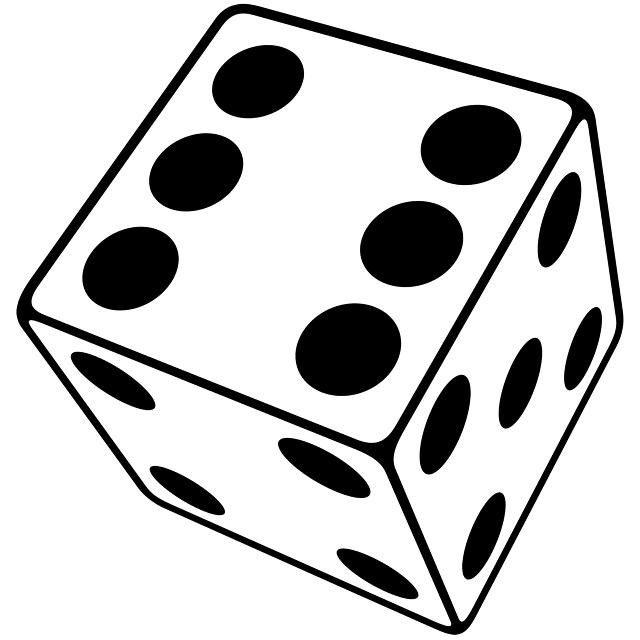
### 概率

概率论研究随机事件。它源于赌徒的研究。赌博中有许多随机事件，比如投掷一个骰子，是否只凭运气呢？

赌徒逐渐发现随机事件的规律。投掷两个骰子是常见的赌博游戏。如果重复很多次，那么总数为2的次数会比总数7的次数少。这就是赌徒把握到的规律：尽管我无法预知事件的具体结果，但我可以了解每种结果出现的可能性。这是概率论的核心。

“概率”到底是什么？这在数学上还有争议。“频率派”认为概率是重复尝试多次，某种结果出现的次数在尝试的总次数的比例。“贝叶斯派”认为概率是主观信念的强弱。幸好，这些争议并不影响我们在日常生活中使用“概率”哲学。天气预报的降雨概率为80%时，很多人会因此带上伞。报纸会分析一场球赛某支球队的赢球概率，如果最终赢球概率为10%的球队取胜，那么球迷会感到惊讶，这毕竟是小概率事件。

要知道某个结果的概率并不容易。上面分析球队的赢球概率，要考虑许多因素。投一个骰子，有6种可能的结果。许多原因会影响到结果，比如撒子是否均匀，比如掷撒子的人是否有技巧偏向。只有在骰子绝对均匀，且没有作弊，每种结果出现的概率才相同。否则的话，根本无法给结果一个确定的概率值。因此，为了能从数学上给结果分配一个概率，我们往往会给随机事件增加一些假设条件。这些条件有理想化的成份，但并不至于偏离现实。比如，我们说掷撒子，撒子均匀，掷的人也没有什么特殊手法，并由此推断每种结果出现的可能相同。那么，其中任意一个结果出现的概率为1/6。



### 基本计数原理

上面我们谈到了“等概率”的假设。如果每种结果出现的概率相同，那么给结果分配概率的任务就变得简单一些。在计算这种概率时，我们只需要等概率的结果的总数，就可以知道每种结果的概率。比如掷一个撒子会有6种结果，如果等概率，那么每个结果的概率为1/6。对于一些复杂的情况，就需要使用到计数技巧。

计数的基本原理叙述如下:

如果一个实验可以分为m个步骤，每个步骤分别有n1,n2,...,nm种可能，那么总共会有

n1×n2×...×nm

种可能的结果。

基本技术原理的核心是“分步”。对于简单的一个步骤的事情，我们能比较直接的分辨结果的总数。比如生一个孩子的性别，比如一个硬币的正反，比如一个撒子的结果。当一个随机事件是多个步骤复合而成的，而每个步骤又都是随机的，那么分布可以简化问题的复杂性。想像一个餐厅，有三个窗口，分别卖三种饮料，五种菜和两种主食。每个学生在每个窗口限选一种，那么学生的餐饮配套会有3x5x2共30种可能的结果。如果每个窗口的师傅都很随意霸道，随手给学生一样东西，那么我们甚至于可以假设等概率条件，每种餐饮拍套出现的概率为1/30。

(当然，作为学生，会抗议这样的“随机”食堂吧？)

基本计数原理的应用并不局限于概率论。在程序员进行算法分析时，无形中使用的就是进行计数。比如嵌套循环，外循环需要M步，内循环需要N步，那么总共进行操作的次数是MxN次。可以说，计数是“离散数学”非常重要的一个组成部分；而离散数学，正是计算机专业的核心数学课程。

基本计数原理是思考的起点。现实中的情况往往会更多变些。特别是当我们“分布”的动作都是作用于同一个群体时，会相对复杂。我们分类了解以下情形:

### 有序的重复抽样

考虑下面的两个问题:

* 一个骰子连续掷2次，所有可能的结果有多少个？
* 一个彩票可选6个号，每个号可以是0到9，共有多少个可能的结果？

我们可以看到，这一类的抽样结果是由多次抽样构成的。每次抽样的样本，在下一次也可能出现。比如骰子第一次为1，第二次还可能为1。这叫做重复抽样 (或者说有放回的抽样，sampling with replacement)。在骰子的例子中，每次抽样的可能出现的结果都有6种。

样本出现的次序影响结果。比如(1,2)和(2,1)是两个不同结果。

从数学上来说，如果进行m次有放回的抽样，每次抽样都有n种可能。如果最终结果有序，那么将有

nm

种可能。

我们下面模拟骰子的例子:

[复制代码](javascript:void(0);)

import itertools

a = [1, 2, 3, 4, 5, 6]

outcomes = list(itertools.product(a, a))

print(outcomes)

print(len(outcomes))

[复制代码](javascript:void(0);)

共返回36个可能的结果:

[复制代码](javascript:void(0);)

[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6),

(2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6),

(3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6),

(4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6),

(5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6),

(6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)]

36

[复制代码](javascript:void(0);)

如果每种结果的出现概率相同，那么对于其中的某个具体结果来说，它出现的概率P=1/36。

### 有序的非重复抽样

考虑下面两个问题:

* 从4个人中，挑出2个人分别担任队长和副队长，有多少种可能？
* 从10们课种，挑选3们，分别放入周一、周三、周五的课表，有多少种可能？

可以看到，这样的抽样是没有重复的。某一次抽样的样本在此后不会出现，前面一个步骤的动作减少了后面一个步骤的选择，这叫做非重复抽样。在非重复的前提下，每次抽样可能的结果数递减，比如从4个人中选一个作为队长，那么副队长只能从3个人中选择。

同样，结果是有序的。A担任队长，B担任副队长，与A担任副队长，B担任队长，是两个不同结果。

有序的非重复抽样又叫做排列(permutation)。从数学上来说，从n个样品中挑选m个，放入m个位置，将有

n×(n−1)×...×(n−m+1)

种可能。如果我们使用阶乘(factorial)运算符，那么结果可以表示为

n!(n−m)!

其中，n!=1×2×...×(n−1)×n。

我们用下面的程序来模拟队长组合的状况:

[复制代码](javascript:void(0);)

import itertools

a = ["Tom", "Lee", "King", "James"]

outcomes = list(itertools.permutations(a, 2))

print(outcomes)

print(len(outcomes))

[复制代码](javascript:void(0);)

结果为

[('Tom', 'Lee'), ('Tom', 'King'), ('Tom', 'James'),

('Lee', 'Tom'), ('Lee', 'King'), ('Lee', 'James'),

('King', 'Tom'), ('King', 'Lee'), ('King', 'James'),

('James', 'Tom'), ('James', 'Lee'), ('James', 'King')]

共有12种可能的结果。

### 无序的非重复抽样

考虑下面的问题:

* 从4个人中抽出2个人，有多少种可能？
* 从一副扑克中抽3张牌，有多少种可能？

在上面的问题中，每次抽样同样是非重复的。但这里，抽样结果是无序的。比如说，抽出"Lee"和"Tom"，以及抽出"Tom"和"Lee"，是同一个结果。这样的抽样方式叫做组合(combination)。

m个样品有m!种排列方式。如果是从n个样品中抽取m个作为组合，所有的这m!种排序方式应该看做一种。因此，有

n!(n−m)!m!

种可能结果。我们可以用下面的方式记录组合:

(nm)=n!(n−m)!m!

我们下面来模拟第一个问题:

[复制代码](javascript:void(0);)

import itertools

a = ["Tom", "Lee", "King", "James"]

outcomes = list(itertools.combinations(a, 2))

print(outcomes)

print(len(outcomes))

[复制代码](javascript:void(0);)

有以下结果

[('Tom', 'Lee'), ('Tom', 'King'), ('Tom', 'James'),

('Lee', 'King'), ('Lee', 'James'),

('King', 'James')]

可以看到，从4个中挑选2个，有6种可能的组合。这是排列的一半。

组合的问题可以进一步延伸。比如，将9个球分为1, 3, 5个的三堆，有多少种方式？这相当于从9个球中抽取1个，再从剩下的8个球中抽取3个，最后剩下的5个为一堆。可以证明，结果为

9!1!3!5!

类似的，将n个球分为n1,n2,...,nm个的堆，其中n=n1+n2+...+nm。将有

n!n1!n2!...nm!

种可能。

### 无序的重复抽样

考虑下面的问题:

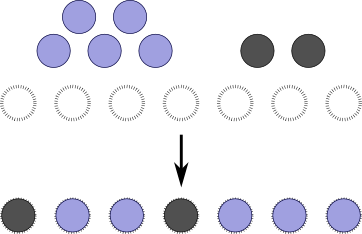
* 刮奖彩票有4种奖品。购买3张彩票的话，有多少种中奖可能？

在上面的每次抽样中，都是重复抽样，即抽出后有放回。比如刮奖中，可以多次刮到同一奖品。我们在一个表中记录结果:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 台灯 | 手表 | 电脑 | 汽车 |
| 可能1 | 3 | 0 | 0 | 0 |
| 可能2 | 2 | 0 | 1 | 0 |
| 可能3 | 0 | 1 | 1 | 1 |

可以看到，我们实际上是将3张彩票分成4份，每份的数目不定(≥0)。

这与下面的问题类似，将5个相同物品放入三个不同的容器中:



[图片来源](http://en.wikibooks.org/wiki/Probability/Combinatorics)

我们用2个黑色分隔物，来将5个相同的物品分为3堆。比如这里，将物品分为(0, 2, 3)的结果。

从7个位置中挑选2个作为分割物的位置，共有

(72)

种可能。

概括来讲，从n个样品中，无序的重复抽样m次，有

(n+m−1m−1)

种可能。

### 阶乘与组合

我们在上面多次使用了阶乘运算，在Python中，它可以使用math.factorial实现:

import math

print(math.factorial(5))

此外，组合可以使用scipy.misc.comb来近似计算，比如:

import scipy.misc

print(scipy.misc.comb(4, 2))

### 练习

每一部分都提出了两个问题，思考第二个问题的结果。

### 总结

基本计数原理

排列

组合

# [概率论02 概率公理](http://www.cnblogs.com/vamei/p/3194360.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

概率论早期用于研究赌博中的概率事件。赌徒对于结果的判断基于直觉，但高明的赌徒尝试从理性的角度来理解。然而，赌博中的一些结果似乎有矛盾。比如掷一个骰子，每个数字出现的概率相等，都是1/6。然而，如果有两个骰子，那么出现的2到12这些数字的概率却不相同。概率论这门学科正是为了搞清楚这些矛盾背后的原理。

早期的概率论是一门混合了经验的数学学科，并没有严格的用语。因此，概率论在数学的精密架构下，显得有些异类。许多名词，如“概率”等，一定程度上是按照人们的直觉来定义的。1933年，俄国数学家Andrei N. Kolmogorov建立了概率论的公理化体系，严格定义了概率论的语言。正如现代数学的其他学科一样，概率论的公理化体系同样基于集合论。公理化的概率论体系基于几条简单易懂的公理，衍生出整个概率论的体系。学习这个公理化的体系，可以消除直觉中的许多混淆。这一公理体系的核心是“概率测度”。

### 实验与样本空间

任何一个过程，如果它的结果是随机的(无法事前知道)，那么该过程就称为一个实验。实验所有可能的结果组成一个集合(set)，叫做样本空间(sample space)，用Ω表示。我们看下面实验的样本空间:

实验1. 连续掷一个硬币两次:

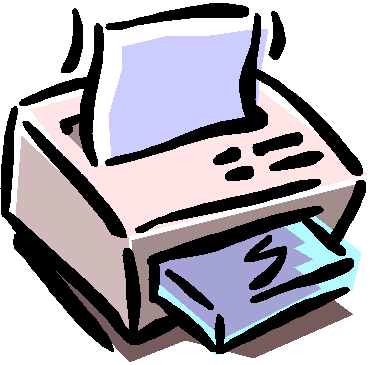
Ω={HH,HT,TH,TT}

H表示正面，T表示反面。上面括号里包含了所有可能的结果：正正，正反，反正，反反。



实验2. 打印机的队列最多允许10个工作。某时刻的工作数目:

Ω={0,1,3,...,9,10}



实验3. 开车经过两个路口，遇到的红绿灯情况:

Ω={rr,rg,ry,gg,gr,gy,yy,yr,yg}

r表示红灯，g表示绿灯，y表示黄灯。

(利用[计数原理](http://www.cnblogs.com/vamei/p/3180875.html)，我们可以知道所有可能结果的总数为9)



对于概率论来说，集合是“如来佛的手掌心”。事实上，整个现代数学体系都是建立在集合论的基础上。集合本身没有什么神秘的，就是一些元素的集合。数学的关键是不同集合的特性、集合内部的结构和集合之间的关系。看似平常的集合给数学带来许多意想不到的结果。

### 事件

样本空间包含了概率论研究的基本元素，也就是实验的结果。它们好象化学里的原子。在掷撒子的游戏中，1，2，3，4，5，6，这些结果就构成了我们的原子。然而，就像赌徒只对“大”和“小”感兴趣一样，在许多时候，我们会对分子那样的原子集合更感兴趣。在概率论里，这样的“分子”就是样本空间的子集。样本空间的一个子集，被称为一个事件(event)。比如说，在实验1中，第一次投掷为正面的所有结果构成子集，即一个事件。该事件包含有两个元素:

A={(H,H),(H,T)}

再比如，第二次投掷为正面也构成一个事件，即

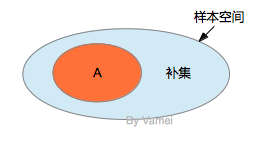
B={(H,H),(T,H)}

我们可以将事件理解为一些特定结果的合集。通过事件，我们可以将结果“聚合”，从而在高一层的单位上进行概率研究。

既然事件是样本空间的一个子集，那么事件可以有补集。事件A的补集包含所有不属于A的样本空间元素。

Ac={TH,TT}

该补集代表的事件为: 第一次投掷是反面。

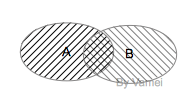


补集

两个集合可以有交集和并集运算。我们以集合A和集合B为例。

C=A∩B

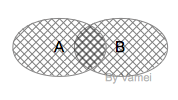
交集C中包含了所有既在A中又在B中的元素。事件C表示第一次为正面且第二次为正面。C={(H,H)}



交集: 交叉阴影区域

D=A∪B

并集D中包含了所有在A中或者在B中的元素。事件D表示第一次为正面或者第二次为正面，D={(H,H),(T,H),(H,T)}



并集: 交叉阴影区域

空集Φ是一个不包含任何元素的集合。如果两个集合的交集为空集，即M∩N=Φ，那么这两个集合不相交。在概率论中，不相交的两个事件互斥。

和加法一样，集合的交并集运算同样有运算法则。这些法则可以如上面那样，画出集合图形，来辅助理解。

交换律

A∪B=B∪A

A∩B=B∩A

结合律

(A∪B)∪C=A∪(B∪C)

(A∩B)∩C=A∩(B∩C)

分配律

(A∪B)∩C=(A∩C)∪(B∩C)

(A∩B)∪C=(A∪C)∩(B∪C)

### 概率测度

我们上面定义了一些基本用语，即“实验”，“样本空间”，“事件”。我们下面要给“分子”上色：引入概率的概念。我们用函数来给每个事件分配一个概率，即分子和颜色的对应关系。

概率测度是基于样本空间Ω的一个函数P。这个函数P定义了从样本空间的子集(即事件)到实数的映射，且满足下面的条件:

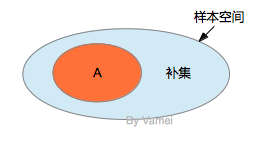
1. P(Ω)=1

2. 如果A⊂Ω, 那么P(A)≥0

3. 如果A1和A2不相交，那么

P(A1∪A2)=P(A1)+P(A2)

“概率测度”是一个有些抽象的概念。“测度”这个词是在提示我们概率定义的基础是“测度论”。粗糙的说，“测度论”用于研究一个集合的“大小”或者说“面积”。更严格的说，就像概率一样，“测度”是集合的子集到实数的一个映射。比如一个正方形的面积为6，实际上是说，一个点的集合（正方形）的某个“测度”为6，即点的集合和实数6对应。“面积”的一个关键特点是可加。比如我们买地的时候，如果两块地不重叠，那么它们的面积总和是两个各自面积的和。概率测度有相同的特点，就是上面的第3点。第1，2两点是概率的基本特征，即所有情况的概率总和为1，而概率值不为负。基于这样一种直观但不严格的类比，我们可以把概率（也就是“概率测度”）想象成“集合的面积”。而“样本空间的总面积为1”。



以上是概率论的公理体系。利用上面的定义以及集合论工具，我们会进一步建立起概率论的体系。但要注意的是，上面公理化的定义，尽管严谨，但并没有说明“概率是什么”，而只是说“概率那个人啊，它应该长的方脸，长鼻子，小眼镜”。这有些像编程中的"duck typing"，也就是根据对象的动作或者特点，来定义对象。即使是今天，概率的本质也存在争议。主流的观点分为两派，即频率观点和贝叶斯观点。在频率观点中，如果我们以相同的条件重复尝试N次，那么如果某个事件出现了n次，那么该事件的概率为P(A)=n/N。在贝叶斯观点中，概率代表了主观上对某一论断的信心。尽管对概率的理解不同，这两个流派都开衍生出了非常有用的工具。

另一方面，定义也没有告诉我们如何确定函数P，即如何计算概率测度。很多时候，函数P的确定依然基于一些假设和一定程度的直觉。比如在等概率条件下，我们利用[计数](http://www.cnblogs.com/vamei/p/3180875.html)方法，来获得概率。比如一枚硬币出现正反两面的概率相同，结果总数为2，那么P(H)=1/2。这也正是我们第一讲中讲解计数的目的所在。然而在其他情况下，比如不均匀硬币，我们不能简单的用1除以结果总数。我们可以利用频率观点，大量重复实验，来获得P函数。

### Python中的集合

集合这一数据结构在多种语言中都有。比如Python中的集合:

[复制代码](javascript:void(0);)

A = set([1, 2, 3, 4])

B = set([3, 4, 5, 6])

print(A & B) # intersection

print(A | B) # union

print(A - B) # difference, element in A, and not in B

print(A ^ B) # symmetric difference, (A | B) - (A & B)

[复制代码](javascript:void(0);)

上面实现了集合的运算。

再比如，我们可以用in来判断元素是否属于集合，以及用>, >=, <, <=来判断两个集合的归属关系，比如一个集合是另一个集合的子集。

A = set([1, 2])

B = set([1, 2, 3])

print(1 in A) # element

print(A < B) # subset

set是一个数据容器，len(), max(), min()函数同样可用于set，分别返回集合中元素总数，集合最大值，集合最小值。此外，set还有一些方法，比如下面的增加和删除元素，注意set中不会有重复的元素:

[复制代码](javascript:void(0);)

A = set([1, 2])

A.add(5) # add an element

print(A)

A.remove(2) # remove an element

print(A)

A.add(1)

print(A) # a set has no repeated elements

[复制代码](javascript:void(0);)

上面的set中元素都为整数，还可以是其他的任意对象。

练习: 利用Python，验证集合的运算律。

### 总结

样本空间，事件

互斥事件

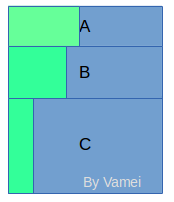
概率测度

# [概率论03 条件概率](http://www.cnblogs.com/vamei/p/3195381.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

在概率公理中，我们建立了“概率测度”的概念，并使用“面积”来类比。这是对概率的第一步探索。为了让概率这个工具更加有用，数学家进一步构筑了“条件概率”，来深入探索概率中包含的数学结构。我们可以考虑生活中常见的一个估计：

三个公司开发一块地。A占地20%，B占地30%，C占地50%。三个公司规划的绿地占比不同：A土地中40%规划为绿地，B土地中的30%规划为绿地，C土地中的10%规划为绿地。我想选择绿地最大的一个小区，应该选择哪一个呢？我们可以画图出来：



显然，我们需要比较的是A：0.2x0.4，B：0.3x0.1，C：0.5x0.1。这是我们常见的一种情形：整个地区分块，每块有一定的比例。再进一步考虑每一块内部的相对比例。我们要了解的“条件概率”这一概念，就对应这里的“相对比例”。

### 条件概率：何弃疗

上面公司的不同造成了绿地占比的不同，也就是说，公司这一因素影响了绿地占比。条件概率同样反映了其它因素对事件概率的影响。

比如说，患者康复有一个概率。在接受治疗和放弃治疗的两种条件下，患者康复的概率也不同。下面是患者的统计结果。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 治疗(T) | 弃疗(NT) | 总数 |
| 康复(R) | 300 | 100 | 400 |
| 未康复(NR) | 200 | 400 | 600 |
| 总数 | 500 | 500 | 1000 |

所有的1000人中，共有400人康复，总体的康复概率为P(R)=400/1000=0.4。另一方面，在接受治疗一列，总共有500人。在这500人种，有300人康复。因此，在接受治疗的条件下，康复的概率变成300/500=0.6。这个概率值高于总体的康复概率。而放弃治疗的条件下，康复的概率为100/500=0.2，康复的概率较低 (可恶，为何放弃治疗)。可见，康复率受到是否接受治疗这一条件的影响。

为了表达某一事件(治疗)对另一个事件(康复)概率的影响，概率论中引入条件概率的概念。条件概率记为P(R|T)=300/500=0.6。R和T是两个事件，即治疗和康复。在治疗(T)的条件下，患者康复(R)的概率为0.6。

(对应文章开始的例子，每个公司的绿地占比为条件概率。比如P(绿地|A公司)=40)



不要放弃治疗啊！

### 条件概率的定义

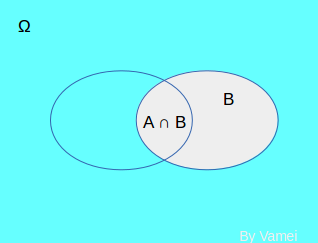
上面给出了条件概率的粗糙概念。但我们已经了解了概率的公理化体系，因此可以基于公理化体系，更严格的定义条件概率。

定义 如果A和B是两个事件，且P(B)≠0。那么B条件下，A的条件概率为

P(A|B)=P(A∩B)P(B)

这是一个非常直观的概念。回到绿地的例子，这里的意思就是说，我们想要知道A公司的绿地占比P(绿地|A公司)的话，可以用A公司占据的绿地面积P(A公司∩绿地)，除以A公司占据土地的面积P(A公司)。

在上面定义条件概率时，我们使用了概率P(A∩B)，即A和B同时发生的概率。从频率的角度上来看，是同时符合A和B的样本数除以Ω中的样本总数。比如上面治疗和康复的例子，P(R∩T)=300/1000。但P(A|B)的隐含假设是，B确定要发生，即病人确定康复。符合这样条件的样本只有500个，而不是整个Ω的1000个样本。



也就是说，当确定B发生时，样本空间不再是Ω，而是缩小成B。我们在B样本空间中寻找A发生的概率。从上面的图中看，就是A∩B的面积(概率测度)，除以B占据的面积(概率测度)，也就是我们条件概率的定义。

### 条件概率的相关推论

条件概率有一些很有用的推论：

推论1 A和B为两个事件，且P(B)≠0。那么

P(A∩B)=P(A|B)P(B)

这个只是将上面的定义中的等式两侧乘以P(B)。从而允许我们从条件概率，来推导两个事件同时发生的概率。

假设卫星观察到，一个地区某一天有云的概率为P(Cloud)=0.2。该地区的地面观测站发现，有云的条件下，当天下雨的为0.5。这是一个条件概率，即P(Rain|Cloud)=0.5。那么既下雨又有云的概率为

P(Cloud∩Rain)=P(Cloud)×P(Rain|Cloud)=0.2×0.5=0.1

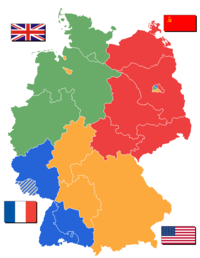


另一个推论，用于通过已知的条件概率，来计算一个事件的概率

推论2 有事件B1,B2,...,Bn。如果⋃ni=1Bi=Ω，两个不同事件互斥(Bi∩Bj=Φ, 如果i≠j），且任意P(Bi)>0。那么，对于任意事件A

P(A)=∑i=1nP(A|Bi)P(Bi)

这个推论的要点是不同的B事件互斥(不相交)，且它们的并集是Ω。每个元素都必须且只能进入一个Bi。在这样的条件下，我们说B1,B2,...,Bn是样本空间的一个分割(partion)。 这就像二战后的德国被分区占领一样，每个Bi是一个占领区。



这正是对所有情况“分块”的思想。再根据每个分块中的某个事件的相对比例，乘以分块自身的权重(“块”的概率)，我们可以求得该事件的绝对占比。

假设家庭收入分为高(H)，中(M)，低(L)三类，高收入家庭占20%，中等收入家庭占65%，低收入家庭占15%。如果高收入家庭的拥有汽车的概率为0.8，中等收入家庭的拥有汽车的概率为0.5，低收入家庭的拥有汽车的概率为0.2。那么任意一个家庭的拥车概率为:

P(C)=P(C|H)P(H)+P(C|M)P(M)+P(C|L)P(L)=0.8×0.2+0.5×0.65+0.2×0.15=0.515



### 独立事件

两个事件可以是相互独立的 (independent)。直观的讲，如果事件A发生与否不会影响事件B的概率，那么A与B独立。

我们尝试将这一个概念用条件概率来表达：将B看作A的条件，那么A的条件概率不受B的影响，即：

定义 两个事件A和B，P(A)≠0，P(B)≠0。如果P(A|B)=P(A)，或者P(B|A)=P(B)，那么事件A和B是独立事件。

某一条件下的“相对占比”等于任意条件下的“绝对占比”？这是怎么一种情况呢？

我们可以想像这样的情况。水中氢和氧的组成比为2：1 (任意条件下)。而水的三种态(水蒸汽、液态水、冰)中的氢和氧组成也是2：1。也就是说，水的态这一条件对氢氧组成无影响，两者独立。

根据独立事件和条件概率的定义可以推知，如果

P(A∩B)=P(A)P(B)

那么A和B独立。

注意，独立事件和互斥事件不同。独立事件是指A发生的概率不影响B。对于互斥事件来说，如果A发生，那么B必然不发生，A的发生影响到了B，所以不是独立事件。比如下雨和不下雨可以看做互斥事件，而下雨和骰子为1可以看做独立事件。

事件[A\_1, A\_2, ..., A\_n]被称为相互独立(mutually independent)，如果对于任意子集[A\_{i\_1},...,A\_{i\_m}]都有

P(Ai1∩...∩Aim)=P(Ai1)...P(Aim)

### 贝叶斯法则

根据上面的定律，我们可以推导出贝叶斯法则(Bayes' Rule)。

贝叶斯法则 如果A和B1,B2,...,Bn为事件，Bi互斥，⋃ni=1Bi=Ω， 且P(Bi)>0。那么

P(Bj|A)=P(A|Bj)P(Bj)∑i=1nP(A|Bi)P(Bi)

这个法则是一种求条件概率的方式。

我们使用文章开头的治疗与康复的例子。我们已知治疗和弃疗的条件概率为P(R|T)=0.6，P(R/NT)=0.2，而P(T)=P(NT)=0.5。

治疗与弃疗互斥(不可能同时治疗又弃疗)，且其并集构成全集(要么治疗，要么弃疗，没有其它的可能)。根据贝叶斯法则，

P(T|R)=P(R|T)P(T)P(R|T)P(T)+P(R|NT)P(NT)=0.6×0.50.6×0.5+0.2×0.5=0.75

即一个康复的人，用药的概率。这与我们在表格中看到的比例相符(400个康复的人中，300个人用药)。

贝叶斯法则常用于求一些比较难以直接获得的条件概率。此外，在机器学习中，也有贝叶斯算法的应用。

练习，编写一个Python函数，用于实现贝叶斯法则的功能。并计算下面的概率:

已知专家预报下雨时，下雨的概率为0.8; 专家预报不下雨时，下雨的概率为0.2。根据以往的经验，专家一年中有30天预报下雨，剩下的天里预报不下雨。问，如果下雨，专家预报的是不下雨的概率为多少?

### 总结

条件概率

独立事件

贝叶斯法则

# [概率论04 随机变量](http://www.cnblogs.com/vamei/p/3196828.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

我们了解了“样本空间”，“事件”，“概率”。样本空间中包含了一次实验所有可能的结果，事件是样本空间的一个子集，每个事件可以有一个发生的概率。概率是集合的一个“测度”。

这一讲，我们将讨论随机变量。随机变量(random variable)的本质是一个函数，是从样本空间的子集到实数的映射，将事件转换成一个数值。根据样本空间中的元素不同(即不同的实验结果)，随机变量的值也将随机产生。可以说，随机变量是“数值化”的实验结果。在现实生活中，实验结果可以是很“叙述性”，比如“男孩”，“女孩”。在数学家眼里，这些文字化的叙述太过繁琐，我们为什么不能拿数字来代表它们呢？

(数学家恐怕是很难成为文学家吧？)



### 离散随机变量

在连续掷两次硬币的例子中，样本空间为:

Ω={HH,HT,TH,TT}

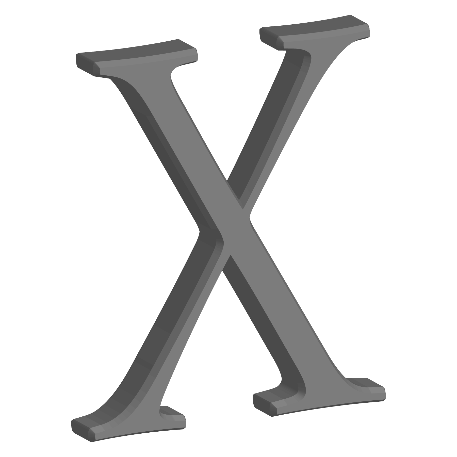
这样的实验结果可以有很多数值化的方法，比如定义HH为400， HT为30， TH为0.2，TT为1。要注意的是，这里是用某个数字来代表样本空间的某个元素，这个数字并不是概率值。

如何对样本空间的元素数值化是根据现实需求的。比如说，根据出现正面的次数，我们将赢取不同的奖励。那么在分析时，可以取“结果中正面的次数”为随机变量。这样一个随机变量将有2, 1, 0三种可能的取值。该随机变量只能取离散的几个孤立值，这样一种随机变量称为离散随机变量。

映射关系如下:

|  |  |
| --- | --- |
| 实验结果 | 随机变量 |
| HH | 2 |
| HT | 1 |
| TH | 1 |
| TT | 0 |

我们通常用一个大写字母来表示一个随机变量，比如X。



如果样本空间中的每个结果等概率，那么随机变量取值可能性为:

P(X=2)=0.25

P(X=1)=0.5

P(X=0)=0.25

当X取0,1,2之外的值时，概率为0。注意到，X=1这个事件，实际上包含了两个元素，HT, TH。因此，X=1出现的概率较高。所有可能取值的概率和为1。

P(X=x)表示了随机变量在不同取值下的概率，称为概率质量函数(PMF, probability mass function)。我们将看到其他的表示概率分布的方式。

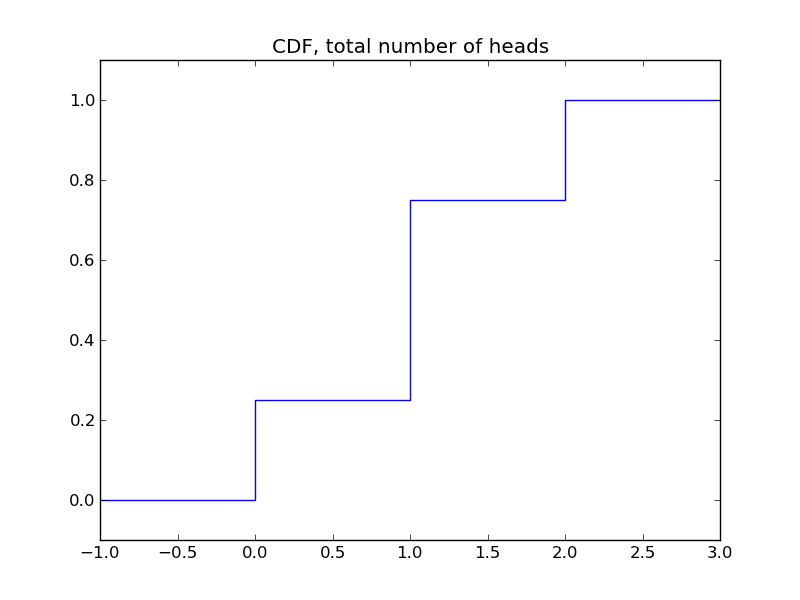
### 累积分布函数

上面的函数列出了每个取值的对应概率。等价的，我们可以用累积分布函数(CDF, cumulative distribution function)来表示随机变量的概率分布状况。在累积分布函数，我们列出的，总是随机变量X，在小于x的这个区间的概率和。当x增大时，X < x包含的结果增加，概率和也相应增加。当x为正无穷时，实际上是所有情况的概率和，那么累积分布函数为1。

严格的定义为:

F(x)=P(X≤x),−∞<x<∞

我们可以绘制上面例子的CDF。



这样的累积分布函数似乎并不比概率质量函数来得方便。但在后面，我们会很快看到它的优势。即它可以同时用于离散随机变量和连续随机变量。

上面的图片可以用如下代码生成:

[复制代码](javascript:void(0);)

# By Vamei

# Plot the CDF of total number of heads in two flips

import matplotlib.pyplot as plt

x = [-1, 0, 0, 1, 1, 2, 2, 3]

y = [0, 0, 0.25, 0.25, 0.75, 0.75, 1.0, 1.0]

fig = plt.figure()

ax = plt.subplot(111)

ax.plot(x, y)

ax.set\_ylim([-0.1, 1.1])

ax.set\_title("CDF, total number of heads")

plt.show()

[复制代码](javascript:void(0);)

### 连续随机变量

随机变量还可以是连续取值，这样的随机变量称为连续随机变量(continuous random variable)。比如，一个随机变量，可以随机的取0到1的任意数值。

当这样取值时，任意区间能实际上都有无穷多个结果。比如，我们测量温度，可以有1度和2度，但两者之间，还可以有1.1度，1.003度，1.658度等等无穷种结果。这样的话，每个结果的可能性都是无穷小。我们讨论的是某个区间内的概率，即P(a<X<b)，而不是具体某一数值的概率。在这样的情况下，分到各个结果的概率都无限趋近于0。显然，我们无法用概率质量函数来描述连续随机变量的分布。

我们这里遇到的困境是现代数学的一个相当的困扰。考虑一个线段，它是点的集合，并且有“长度”这样的测度。然而，线段上有无穷个多个点。讨论“每个点的长度”是完全没有意义的。将线段换成区间，将点换成取值，将长度换成概率，我们发现这两个问题异常相似。另一方面，我们知道，可以从线段上截取某一小段，而这一小段是可以有“长度”的。连续随机变量的概率定义，正依赖于此：对于连续随机变量，我们只讨论某个区间，比如从1.2到1.4这一区间的概率，而不讨论具体某个点，比如1.3的概率。



观察一个很简单的连续随机分布。假设我们有一个随机数生成器，产生一个从0到1的实数，每个实数出现的概率相等。这样的一个分布被称为均匀分布(uniform distribution)。直觉告诉我们，相同长度的每一段区间，对应的概率都相同。由此，[0, 0.5]是整个区间的一半，概率为1/2。对于均匀分布来说，概率正好和区间长度这一测度等同。

我们尝试用更正式的方式来描述分布。累积分布函数本身就表示随机变量在一个区间概率，所以可以直接用于连续随机变量。即

F(x)=P(X≤x),−∞<x<∞

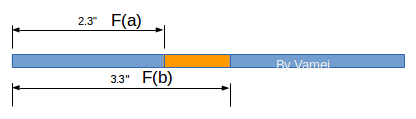
对于均匀分布来说，它的累积分布函数是:

F(x)=0,x<0

F(x)=x,0≤x≤1

F(x)=1,x>1

它类似从线段的一头到某一点的“长度”。这样，我们就知道了从起点到每一点的长度。如果我们想知道某个特定区间[a, b]的概率，它就是F(b) - F(a)。



借用“无穷小”的概念，我们可以构建概率密度函数(PDF，probability density function)。粗糙的讲，我们在某个点附近取一个“无穷小”段，该小段的区间长度为dx，而这个“无穷小”段对应的概率为dF，那么该点的概率密度为dF/dx。这实际上是微积分的领域。

概率密度函数可以代替累积分布函数，来表示一个连续随机变量的概率分布:

f(x)=dF(x)dx

即密度函数是累积分布函数的微分，或者说，

F(x)=∫x−∞f(u)du

即累积分布函数是密度函数从负无穷到x的积分。

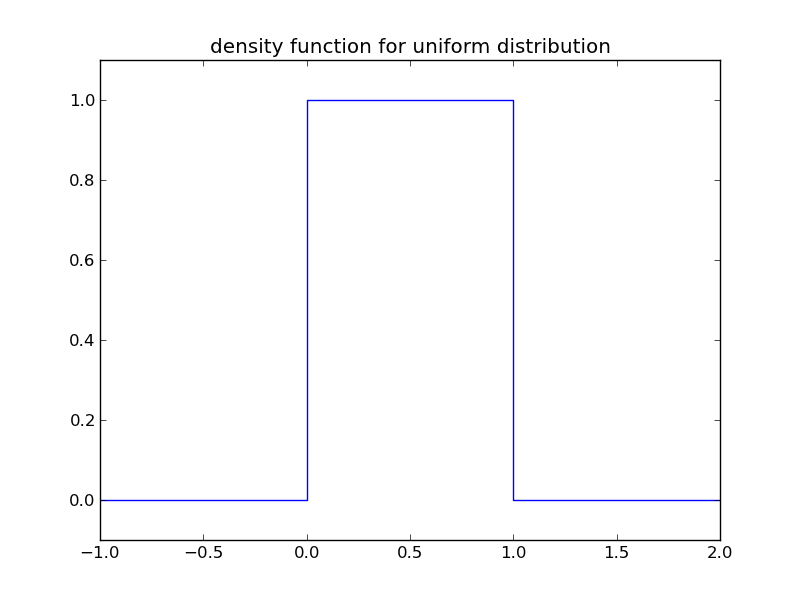
密度函数满足:

∫+∞−∞f(u)du=1

均匀分布的密度函数可以写成:

f(x)={1,0≤x≤10,x<0orx>1

可以画出该密度函数



对一个函数的积分，获得的是该函数曲线下的面积。因此，密度曲线下某个区间的面积，就是密度概率函数的积分，代表了随机变量在该区间的概率。概率密度函数就可以非常直观的通过“面积”，来表示概率的大小。

从负无穷到正无穷积分，就代表了所有可能结果的概率和，即为1。

上面的图片可以利用下面代码生成:

[复制代码](javascript:void(0);)

# By Vamei

# Density function for uniform distribution

import matplotlib.pyplot as plt

x = [-1, 0, 0, 1, 1, 2]

y = [0, 0, 1, 1, 0, 0]

fig = plt.figure()

ax = plt.subplot(111)

ax.plot(x, y)

ax.set\_xlim([-1, 2])

ax.set\_ylim([-0.1, 1.1])

ax.set\_title("density function for uniform distribution")

plt.show()

[复制代码](javascript:void(0);)

### 总结

随机变量，随机变量的概率分布

累积分布函数

密度函数

# [概率论05 离散分布](http://www.cnblogs.com/vamei/p/3198371.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

我们已经知道什么是离散随机变量。离散随机变量只能取有限的数个离散值，比如投掷一个撒子出现的点数为随机变量，可以取1，2，3，4，5，6。每个值对应有发生的概率，构成该离散随机变量的概率分布。

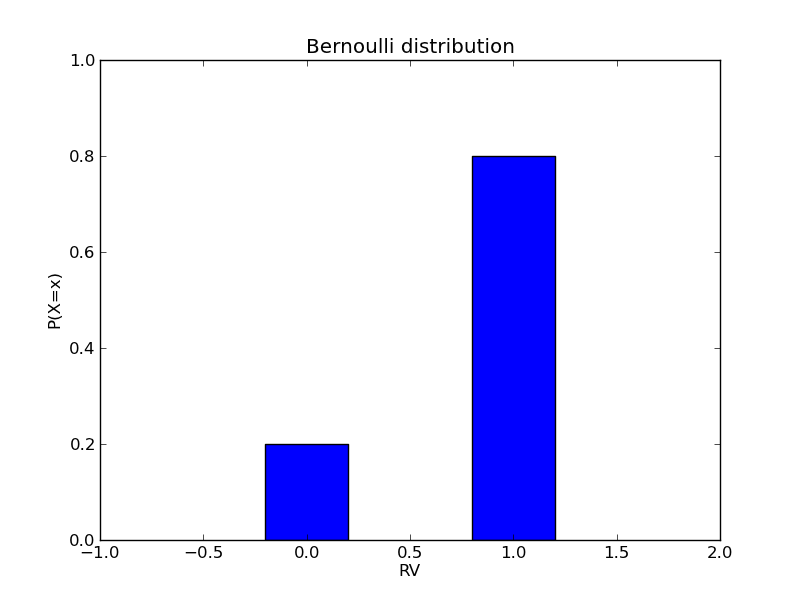
离散随机变量有很多种，但有一些经典的分布经常重复出现。对这些经典分布的研究，也占据了概率论相当的一部分篇幅。我们将了解一些离散随机变量的经典分布，了解它们的含义和特征。

### 伯努利分布

伯努利分布(Bernoulli distribution)是很简单的离散分布。在伯努利分布下，随机变量只有两个可能的取值: 1和0。随机变量取值1的概率为p。相应的，随机变量取值0的概率为1-p。因此，伯努利分布可以表示成:

P(X=k)={p1−pforfork=1k=0

投掷一次硬币，出现正面，记录为1，出现背面，记录为0。这样我们就有一个伯努利随机变量。如果硬币是均匀的，那么p=0.5。如果硬币是不均匀的，比如硬币出现正面的概率为0.8，那么p=0.8。我们可以绘制此分布如下:



代码如下:

[复制代码](javascript:void(0);)

# By Vamei

plt.bar([-0.2,0.8],[0.2, 0.8],width=0.4) # bar plot

plt.xlim([-1, 2]) # axis range

plt.ylim([0.0, 1.0])

plt.title("Bernoulli distribution") # figure title

plt.xlabel("RV")

plt.ylabel("P(X=x)")

plt.show()

[复制代码](javascript:void(0);)

在scipy.stats中，有直接表达伯努利分布的函数bernoulli。事实上，在scipy.stats中，有许多常见的分布函数。

# By Vamei

from scipy.stats import bernoulli

rv = bernoulli(0.8)

x = [-1, 0, 1, 2]

print(rv.cdf(x))

上面，我们创建了一个p=0.8的伯努利随机变量，并计算该随机变量在不同点的累积分布函数(CDF)。

### 二项分布

为了理解二项分布是如何出现的，我们假设下面情况：进行n次独立测试，每次测试成功的概率为p(相应的，失败的概率为1-p)。这n次测试中的“成功次数”是一个随机变量。这个随机变量符合二项分布(binomial distribution)。

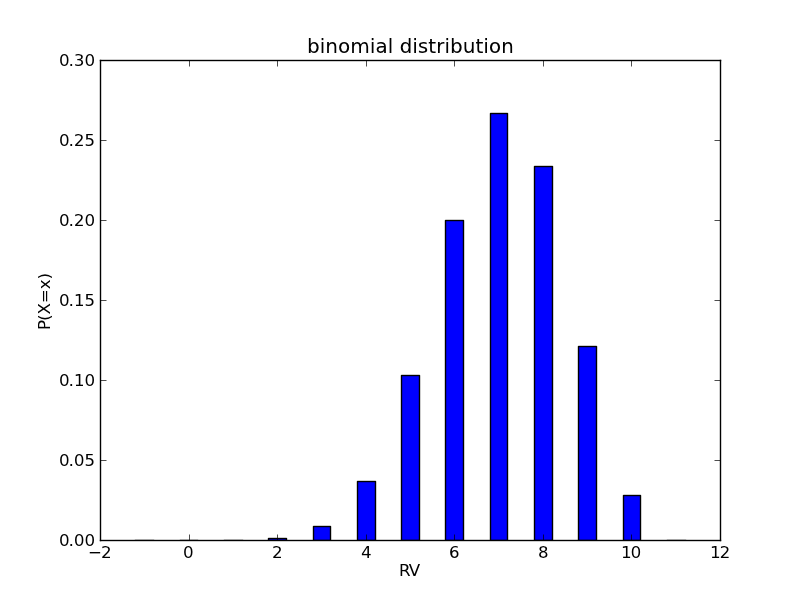
二项分布可以从计数的角度来理解。n次测试，如果随机变量为k，意味着其中的k次成功，n-k次失败。从n次实验中挑选k个，根据计数原理，共有(nk)种可能。其中的每种可能出现的概率为pk(1−p)n−k)。因此，二项分布可以表示成为:

P(X=k)=(nk)pk(1−p)n−k

k=0,1,2,...,n

(“二项分布”的命名原因是，上面的P(X=k)等于二项式(p+(1−p))n)二项式展开的第k项。)

我们进行连续的10次打靶，如果每次中靶的概率为0.7， 那么在10次打靶中，打中靶的次数就是一个符合二项分布的随机变量。在这样的假设下，n=10，p=0.7，k可以取值从0到10之间的任意整数。利用scipy.stats中的binom函数，我们可以绘制此分布如下:



x=0和x=1概率不为0，只是值太小，没有在图中显现出来。

代码如下:

[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import binom

rv = binom(10, 0.7)

x = np.arange(-1, 12, 1)

y = rv.pmf(x)

plt.bar(x-0.2, y, width=0.4)

plt.title("binomial distribution")

plt.xlabel("RV")

plt.ylabel("P(X=x)")

plt.show()

[复制代码](javascript:void(0);)

### 泊松分布



泊松

泊松分布(Poisson distribution)是二项分布的一种极限情况，当p→0,n→+∞，而np=λ时，二项分布趋近于泊松分布。这意味着我们进行无限多次测试，每次成功概率无穷小，但n和p的乘积是一个有限的数值。

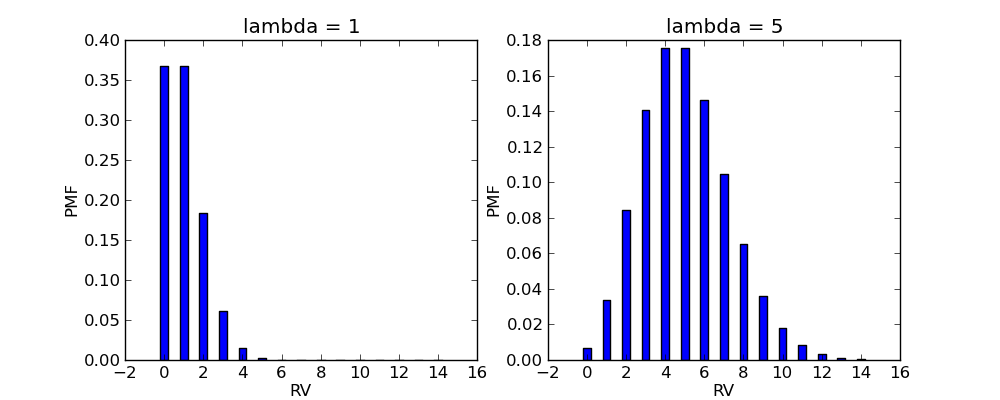
泊松分布用于模拟低概率事件，比如地震。地震是很低概率的事件，我们想知道一段时间，比如十年内某地发生地震的总数，可以将十年划分为n个小时间段，每个时间段内地震发生的概率为p。我们假设小时间段很短，以致于不可能有两次地震发生在同一小时间段内，那么地震的总数是一个随机变量，趋近于泊松分布。

泊松分布的关键特征是，随机变量的取值与区间的长短成正比。这里的区间是广义的，它既可以表示时间，也可以表示空间。泊松分布有一个参数λ，我们可以将泊松分布写成如下形式:

P(X=k)=λkk!e−λ

k=0,1,2,...

λ=1和λ=5的泊松分布如下:



可以看到，λ决定了泊松分布的“重心”所在。比如地震的例子中，λ越大，k取大值的可能性越大，越有可能发生更多次的地震。我们将在统计中看到，如何利用观测的数据，来估计λ的取值。

代码如下:

[复制代码](javascript:void(0);)

# By Vamei

# use poisson function

from scipy.stats import poisson

rv1 = poisson(1)

x = np.arange(0,15)

y1 = rv.pmf(x)

plt.figure(figsize=(10, 4))

plt.subplot(121)

plt.bar(x-0.2, y1, width=0.4)

plt.title("lambda = 1")

plt.xlabel("RV")

plt.ylabel("PMF")

plt.subplot(122)

rv2 = poisson(5)

y2 = rv2.pmf(x)

plt.bar(x-0.2, y2, width=0.4)

plt.title("lambda = 5")

plt.xlabel("RV")

plt.ylabel("PMF")

plt.show()

[复制代码](javascript:void(0);)

### 几何分布

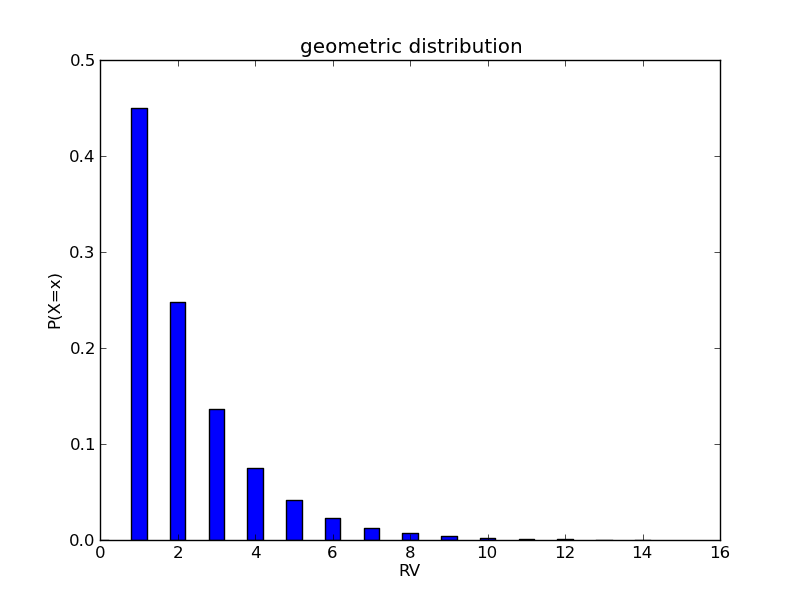
假设我们连续进行独立测试，直到测试成功。每次测试成功的概率为p。那么，到我们成功时，所进行的测试总数是一个随机变量，可以取值1到正无穷。这样一个随机变量符合几何分布(geometric distribution)。

随机变量取值为k时，意味前面的k-1次都失败了。因此，我们可以将几何分布表示成:

P(X=k)=(1−p)k−1p

k=1,2,...

假设我们进行产品检验。产品的合格率为0.65。我们需要检验k次才发现第一个合格产品，k的分布表示如下:



可以看到，几何分布的概率质量函数呈递减趋势。我们也可以从表达式中得到该特征。

代码如下:

[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import geom

rv = geom(0.45)

x = np.arange(-1, 15, 1)

y = rv.pmf(x)

plt.bar(x-0.2, y, width=0.4)

plt.ylim([0, 0.5])

plt.title("geometric distribution")

plt.xlabel("RV")

plt.ylabel("P(X=x)")

plt.show()

[复制代码](javascript:void(0);)

### 负二项分布

几何分布实际上是负二项分布(negative geometric distribution)的一种特殊情况。几何分布是进行独立测试，直到出现成功，测试的总数。负二项分布同样是进行独立测试，但直到出现r次成功，测试的总数k。r=1时，负二项分布实际上就是几何分布。

在连续的r次测试时，我们只需要保证最后一次测试是成功的，而之前的k-1次中，有r-1次成功。这r-1次成功的测试，可以任意存在于k-1次测试。因此，负二项分布的表达式为:

P(X=k)=(k−1r−1)pr(1−p)k−r

k=1,2,...

练习: (可以使用scipy.stats中的ngeom函数来表示负二项分布) 假设我们进行产品检验。产品的合格率为0.65。我们需要检验k次才共发现3个合格产品。绘制随机变量k的概率分布。

### 超几何分布

一个袋子中有n个球，其中r个是黑球，n-r是白球，从袋中取出m个球，让X表示取出球中的黑球的个数，那么X是一个符合超几何分布(hypergeometric distribution)的随机变量。

练习: 推导超几何分布的概率质量函数，并绘制其概率分布。

### 总结

离散随机变量比较直观，容易理解。我们在这里介绍了一些经典分布，即随机变量取值的概率。

# [概率论06 连续分布](http://www.cnblogs.com/vamei/p/3199522.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

在[随机变量](http://www.cnblogs.com/vamei/p/3196828.html)中，我提到了连续随机变量。相对于离散随机变量，连续随机变量可以在一个连续区间内取值。比如一个均匀分布，从0到1的区间内取值。一个区间内包含了无穷多个实数，连续随机变量的取值就有无穷多个可能。

为了表示连续随机变量的概率分布，我们可以使用累积分布函数或者密度函数。密度函数是对累积分布函数的微分。连续随机变量在某个区间内的概率可以使用累积分布函数相减获得，即密度函数在相应区间的积分。

在[随机变量](http://www.cnblogs.com/vamei/p/3196828.html)中，我们了解了一种连续分布，即均匀分布(uniform distribution)。这里将罗列一些其他的经典连续分布。

### 指数分布

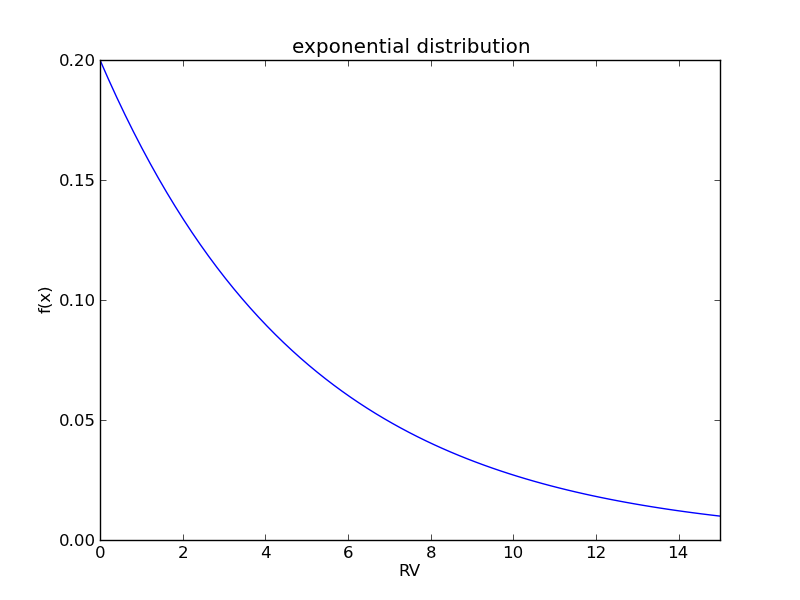
指数分布(exponential distribution)的密度函数随着取值的变大而指数减小。指数分布的密度函数为:

f(x)={λe−λx0ififx≥0x<0

累积分布函数为:

F(x)=1−e−λx,x≥0

我们绘制一个指数分布λ=0.2，如下:



这样一种分布在生活中很常见。比如，洪水等级的分布就类似于这样一个分布。小等级的洪水常发生，而大洪水发生的概率则很小。再比如，金矿的分布：大部分矿石的含金量少，而少部分矿石的含金量高。这提醒我们，一些特殊的条件导致了指数分布。感兴趣的话可以学习“随机过程”这一数学分支。

代码如下:

[复制代码](javascript:void(0);)

from scipy.stats import expon

import numpy as np

import matplotlib.pyplot as plt

rv = expon(scale = 5)

x = np.linspace(0, 20, 100)

plt.plot(x, rv.pdf(x))

plt.xlim([0, 15])

plt.title("exponential distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

上面的expon函数接收一个参数scale。参数scale等于1/λ

指数分布是无记忆(memoryless)的。我们以原子衰变为例。任意时刻往后，都需要10年的时间，会有一半的原子衰变。已经发生的衰变对后面原子衰变的概率分布无影响。用数学的语言来说，就是

P(X>s)=P(X>s+t|X>t),fors,t≥0

等式的左边是原子存活了s的概率。而等式的右边是某一时刻t之后，原子再存活s时间的概率。可以利用指数分布的累积分布函数，很容易的证明上面的等式。指数分布经常用于模拟人的寿命或者电子产品的寿命，这意味着我们同样假设这些分布是无记忆的。一个人活10年的概率和一个人到50岁后，再活10年的概率相等。这样的假设有可能与现实情况有所出入，需要注意。

### 正态分布

正态分布(normal distribution)是最常用到的概率分布。正态分布又被称为高斯分布(Gauss distribution)，因为高斯在1809年使用该分布来预测星体位置。吐槽一句，第一个提出该分布的并不是数学王子高斯，而是法国人De Moivre。作为统计先驱，这位数学家需要在咖啡馆“坐台”，为赌徒计算概率为生。(看来法国咖啡馆不止有文艺青年，也有技术屌丝啊。)



 Abraham De Moivre



Gauss

正态分布的发现来自于对误差的估计。早期的物理学家发现，在测量中，测量值的分布很有特点：靠近平均值时，概率大；远离平均值时，概率小。比如我们使用尺子去测量同一个物体的长度，重复许多次。如果没有系统误差，那么测量到的长度值是一个符合正态分布的随机变量。再比如，在电子信号中白噪音，也很有可能符合正态分布。De Moivre最早用离散的二项分布来趋近这一分布，而高斯给出了这一分布的具体数学形式。

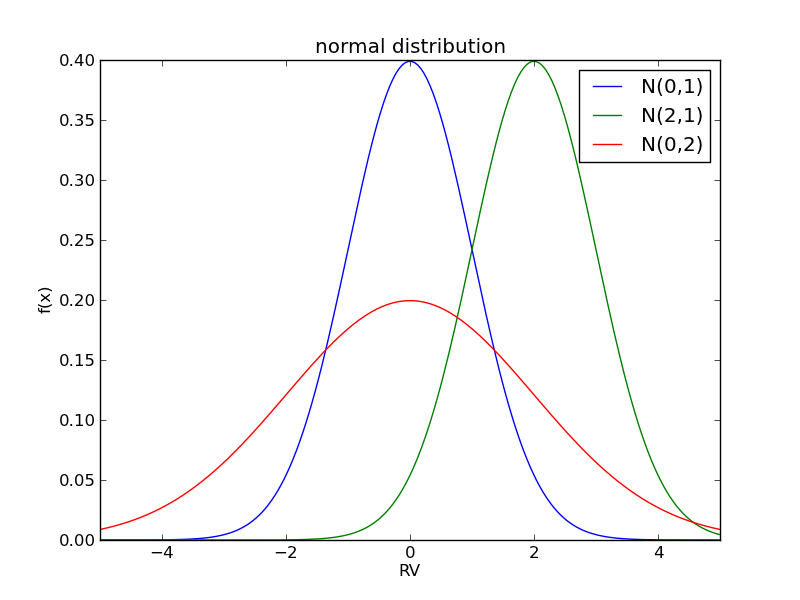
正态分布自从一出生就带着无比强大的“主角光环”，它的特殊地位在后面文章中的中心极限定理中凸显出来。

正态分布的密度函数如下:

f(x)=12π−−√σe−(x−μ)2/2σ2,−∞<x<∞

正态分布有两个参数，μ和σ。我们可以将正态分布表示成N(μ,σ)。当μ=0，σ=1，这样的正态分布被称作标准正态分布(standard normal distribution)。

我们绘制三个正态分布的密度函数:



可以看到，正态分布关于x=μ对称，密度函数在此处取得最大值，并随着偏离中心而递减。如果以测量长度为例，这说明的读取值靠近μ的可能性较大，而偏离μ的可能性变小。

σ代表了概率分布的离散程度。σ越小，概率越趋近对称中心x=μ。

代码如下:

[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import norm

import numpy as np

import matplotlib.pyplot as plt

rv1 = norm(loc=0, scale = 1)

rv2 = norm(loc=2, scale = 1)

rv3 = norm(loc=0, scale = 2)

x = np.linspace(-5, 5, 200)

plt.plot(x, rv1.pdf(x), label="N(0,1)")

plt.plot(x, rv2.pdf(x), label="N(2,1)")

plt.plot(x, rv3.pdf(x), label="N(0,2)")

plt.legend()

plt.xlim([-5, 5])

plt.title("normal distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

正态分布在统计中有非常重要的地位。我们将在后面的中心极限理论的讲解中，看到这一点。

### Gamma分布

Gamma分布在统计推断中具有重要地位。它的密度函数如下:

g(t)=λαΓ(α)tα−1e−λt,t≥0

其中的Gamma函数可以表示为:

Γ(x)=∫0∞ux−1e−udu,x>0

注意到，Gamma分布有两个控制参数α和λ。

练习，利用scipy.stats.gamma绘制α=1,λ=1和α=5,λ=1的Gamma分布密度函数。

### 总结

我们研究了三种连续随机变量的分布，并使用概率密度函数的方法来表示它们。密度函数在数学上比较容易处理，所以有很重要的理论意义。

密度函数在某个区间的积分，是随机变量在该区间取值的概率。这意味着，在密度函数的绘图中，概率是曲线下的面积。

# [概率论07 联合分布](http://www.cnblogs.com/vamei/p/3224111.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

我之前一直专注于单一的随机变量及其概率分布。我们自然的会想将以前的结论推广到多个随机变量。联合分布(joint distribution)描述了多个随机变量的概率分布，是对单一随机变量的自然拓展。联合分布的多个随机变量都定义在同一个样本空间中。

对于联合分布来说，最核心的依然是概率测度这一概念。

### 离散随机变量的联合分布

我们先从离散的情况出发，了解多个随机变量并存的含义。

之前说，一个随机变量是从样本空间到实数的映射。然而，所谓的映射是人为创造的。从一个样本空间，可以同时产生多个映射。比如，我们的实验是连续三次投硬币，样本空间为

Ω={hhh,hht,hth,thh,htt,tht,tth,ttt}

h为正面，t为反面。在同一样本空间上，我们可以定义多个随机变量，比如:

* X: 投掷为正面的总数，可以取值0，1，2，3
* Y: 最后一次出现负面的总数，可以取值0，1
* Z: 将正面记为10，负面记为5，第一次与第三次取值的差，可以有5, -5, 0

这三个随机变量可以看作一个有三个分量的矢量。所以定义在同一样本空间的多随机变量，是一个从样本空间到矢量的映射。

(从这个角度上说，单一随机变量是一个从样本空间到一个有一个分量的矢量的映射)

如果样本空间Ω中每个结果出现的概率相等。而样本空间中共有8个结果，那么个每个结果的出现的概率都是1/8。据此，我们可以计算联合概率，比如

P(X=0,Y=1)=P({ttt})=1/8

P(X=1,Y=1)=P({htt,tht})=2/8

对于X=x,Y=y，我们寻找样本空间中满足这两个取值的所有元素。这些元素构成一个样本空间的子集，该子集的概率就是P(X=x,Y=y)的联合概率。p(x,y)=P(X=x,Y=y)称为联合概率质量函数(joint PMF, joint probability mass function)。联合概率可以看做两个事件同时发生时的概率，事件A为X=x，事件B为Y=y，即P(A∩B)。

找到所有可能取值组合的概率，就找到了这两个随机变量的联合分布:

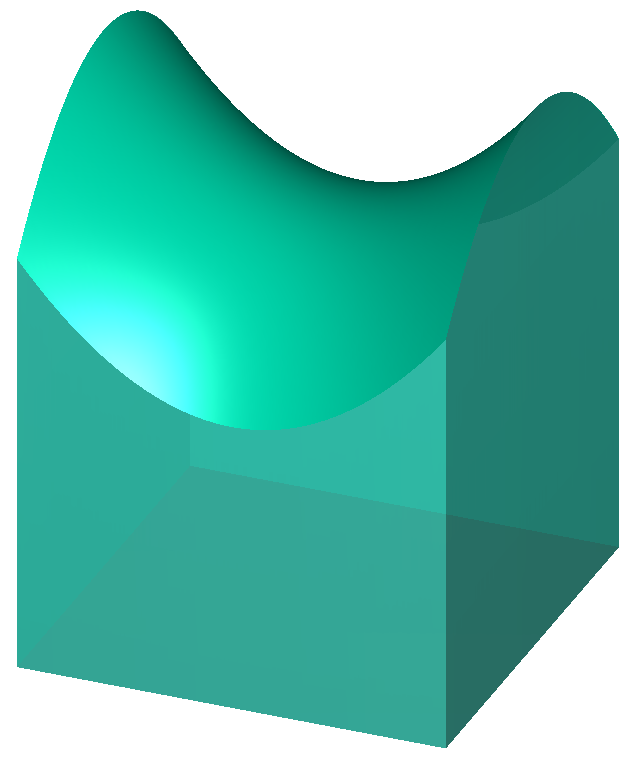
|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | P(X,Y) | 对应子集 |
| 0 | 0 | 0 | Φ |
| 1 | 0 | 1/8 | tth |
| 2 | 0 | 2/8 | thh, hth |
| 3 | 0 | 1/8 | hhh |
| 0 | 1 | 1/8 | ttt |
| 1 | 1 | 2/8 | htt, tht |
| 2 | 1 | 1/8 | hht |
| 3 | 1 | 0 | Φ |

 联合分布

联合分布描述了所有可能的取值情况。因此，联合概率密度函数的累积和为1。

### 连续随机变量的联合分布

我们知道，单个连续随机变量的概率是变量在某个区间(某段线的“长度”)取值的概率。做类似的推广，多个连续随机变量的概率，是这多个随机变量在多维区间的概率。比如两个随机变量，我们需要表达一个二维区间的概率，比如P(a≤X≤b,c≤Y≤d)。这个二维区间可以有一个类似于一个小补丁的“面积”。二维区间对应的概率是一个体积。



面积对应的体积

在单变量情况下，概率是一个“面积”，是由区间的“长度”和密度函数(一条曲线)围成的。这里的“体积”是二维区间的“面积”和密度函数(一个曲面)围成的。我们可以使用联合概率密度函数(joint PDF, joint probability density function)来表达多随机变量的分布。对于双变量的联合分布来说，它等于无穷小块的概率，除以无穷小块的面积。

用微积分的语言来说，就是

P(a≤X≤b,c≤Y≤d)=∫ba∫dcf(x,y)dxdy

f(x,y)就是描述X和Y的联合分布的联合概率密度函数。

联合概率密度函数描述了所有可能取值的情况，因此有

∫+∞−∞∫∞−∞f(x,y)dxdy=1

**实例**

下面是两个连续随机变量的联合PDF:

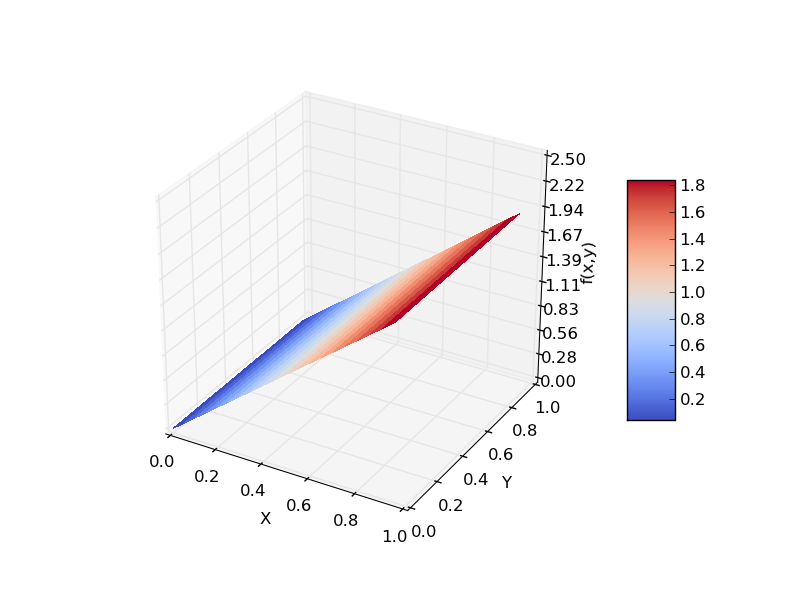
f(x,y)={2x0forfor0≤x,y≤1else

通过积分，计算X在0到0.5，而Y在0到1的概率:

P(0≤X≤0.5,0≤Y≤1)=∫0.50∫102xdxdy=0.25

我们之间也说到，单个随机变量的概率对应线段到概率密度曲线之间的面积。而两个随机变量的概率对应小块到概率密度面之间的体积。

我们可以绘制f(x,y)的分布图形，即一个二维的平面。图中的颜色标记了f(x, y)值的大小。如下:

可以看到，f(x, y)随X增大而增大，在X值确定时，f(x, y)不随Y变化。

[复制代码](javascript:void(0);)

# By Vamei

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib import cm

from matplotlib.ticker import LinearLocator, FormatStrFormatter

import matplotlib.pyplot as plt

import numpy as np

fig = plt.figure()

ax = fig.gca(projection='3d')

X = np.arange(0, 1, 0.05)

Y = np.arange(0, 1, 0.05)

X, Y = np.meshgrid(X, Y)

Z = 2\*X

surf = ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.coolwarm,

linewidth=0, antialiased=False)

ax.set\_zlim(0.0, 2.5)

ax.zaxis.set\_major\_locator(LinearLocator(10))

ax.zaxis.set\_major\_formatter(FormatStrFormatter('%.02f'))

ax.set\_xlabel("X")

ax.set\_ylabel("Y")

ax.set\_zlabel("f(x,y)")

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()

[复制代码](javascript:void(0);)

### 边缘概率

联合分布包含了多个随机变量的分布信息。我们当然可以从联合分布中，提取出任意一个单一随机变量的分布，也就是所谓的边缘分布(marginal distribution)。

对于离散随机变量，可以获得边缘概率质量函数(marginal pmf):

pX(x)=∑allyp(x,y)

pY(y)=∑allxp(x,y)

在求X的单一边缘分布时， 我们累加了相同x值、不同y值时的多个联合概率，从而获得该x值的的总体概率，即边缘概率。

连续随机变量X的边缘密度函数(marginal pdf, marginal probability density function)可以定义为

fX(x)=∫+∞−∞f(x,y)dy

fX(x)是联合密度函数对Y的积分。通过积分，我们将不同Y取值时的联合概率加在一起，就获得纯粹的单一X的分布状况。

类似的，Y的边缘密度函数为

fY(y)=∫+∞−∞f(x,y)dx

取离散随机分布的例子，即掷三次硬币

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | p(y) |
| 0 | 0 | 1/8 | 2/8 | 1/8 | 1/2 |
| 1 | 1/8 | 2/8 | 1/8 | 0 | 1/2 |
| p(x) | 1/8 | 3/8 | 3/8 | 1/8 |  |

边缘概率是对各行和列的累加。最后一列p(y)是Y的分布，Y有1/2的概率取0，1/2的概率取1。最后一行p(x)是X的分布。

取连续随机分布的例子，即下面的连续分布:

f(x,y)={2x0forfor0≤x,y≤1else

可以得到:

fX(x)=2x,0≤x≤1

fY(y)=1,0≤y≤1

### 条件分布

我们之前基于事件介绍了条件概率，即如果事件B发生，那么事件A发生的概率。相同的概念可以引申到随机变量。随机变量取某个值，这可以看做一个事件。我们想知道，随机变量Y取值y，另一个随机变量X为x的概率。

与[事件的条件概率](http://www.cnblogs.com/vamei/p/3195381.html)类似，假设pY(y)≠0，在Y=y的条件下，随机变量X取值为x的概率定义为:

p(x|y)=p(x,y)pY(y)

即X=x,Y=y同时发生的概率，除以Y取值为y的的概率。

以掷三次硬币为例。条件为Y值取值0，即最后一次投掷为正面时。此时，X取值为2有两种可能，即前两次为ht和th。由于前两次投掷有四种组合，所以概率为0.5。

我们可以通过条件概率的公式计算并验证:

p(2|0)=p(2,0)pY(0)=2/81/2=0.5

如果说概率是分一个总和为1的大饼，如果大饼分八块，每块就是1/8。假设半个饼上撒胡椒，另半个饼上撒辣椒。那么在胡椒饼(相当于我们的条件)上选取一块的概率，就是1/4。此时，也就是用原来的概率除以胡椒饼所占的比重。

对于连续随机变量，假设fY(y)≠0，给定Y=y，随机变量X的条件分布为:

f(x|y)=f(x|Y=y)=f(x,y)fY(y)

### 独立随机变量

正如事件之间可以相互独立一样，随机变量之间也可以相互独立。当X独立于Y时，我们可以相像，Y的取值，将不影响X的概率。也就是说

p(x|y)=pX(x)

这意味着，当且仅当

p(x,y)=pX(x)pY(y)

时，X和Y相互独立。

可以验证，连续投掷三次硬币的例子中，X和Y并不独立，比如

p(1,1)=2/8

pX(1)=3/8

pY(1)=1/2

因此，

p(1,1)≠pX(1)pY(1)

X和Y并不独立。

对于连续随机变量来说，当且仅当

f(x,y)=fX(x)fY(y)

时，X和Y相互独立。

对于分布

f(x,y)={2x0forfor0≤x,y≤1else

使用之前获得的边际分布，可以验证

f(x,y)=fX(x)fY(y)

因此，对于该分布来说，X和Y相互独立。

### 总结

通过联合分布，我们将单随机变量的分布拓展到多随机变量的分布。同样的，在单随机变量中引入的条件概率，也可以使用到多随机变量。我们还探讨了随机变量的独立性。

# [概率论08 随机变量的函数](http://www.cnblogs.com/vamei/p/3226755.html)

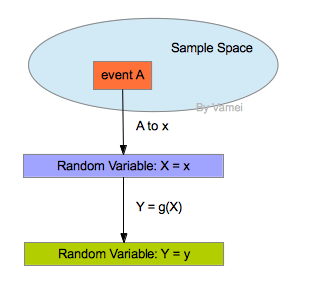
作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

### 随机变量的函数

在前面的文章中，我先将概率值分配给各个事件，得到事件的概率分布。

通过事件与随机变量的映射，让事件“数值化”，事件的概率值转移到随机变量上，获得随机变量的概率分布。

我们使用随机变量的函数，来定制新的随机变量。随机变量的函数是从旧有的随机变量到一个新随机变量的映射。通过函数的映射功能，原有随机变量对应新的随机变量。通过原有随机变量的概率分布，我们可以获知新随机变量的概率分布。事件，随机变量，随机变量函数的关系如下:



一个简单的例子是掷硬币。出现正面的话，我赢1个筹码，负面的话，我输1个筹码。那么，投掷一次，赢的筹码数是一个随机变量X，X可能取值为1和-1。因此X的分布为:

P(1)=0.5

P(−1)=0.5

换一个角度来思考，我们将正负面“换算”成输赢的钱。如果一个筹码需要10元钱买，那么投掷一次硬币，赢的钱是一个随机变量Y，且Y=10X。Y的分布为:

P(10)=0.5

P(−10)=0.5

Y实际上是随机变量X的一个函数。X的1对应Y的10，X的-1对应Y的-10。即Y=10X

小总结，在上面的实验中，硬币为正面为一个事件。赢得的筹码数为一个随机变量X。赢得的钱是X的函数Y，它也是一个随机变量。

随机变量的函数还可以是多变量函数，Y=g(X1,X2,...,Xn)。Y的值y对应的是多维空间的点(x1,x2,...,xn)。比如掷硬币，第一次赢的筹码为X1，第二次赢的筹码为X2。我们可以构成一个新的随机变量Y=X1+X2，即两次赢得的筹码的总和。

### 获得新概率分布的基本方法

一个核心问题是，如何通过X的概率分布，来获得Y=g(X)的概率分布。基本的思路是，如果我们想知道Y取某个值y的概率，可以找到对应的X值x的概率。这两个概率相等。

因此，我们使用如下方法来获得Y的概率。如果有函数关系Y=g(X1,X2,...,Xn)，获得Y分布的基本方法是:

1. 通过Y=g(X1,X2,...,Xn)，找到对应{Y≤y}的(x1,x2,...,xn)区间I。

2. 在区间I上，积分f(x1,x2,...,xn)，获得P(Y≤y)

3. 通过微分，获得密度函数。

如果有函数关系Y=X2， 而X满足下面的分布:

f(x)=12π−−√e−x2/2

对于任意y≥0来说，

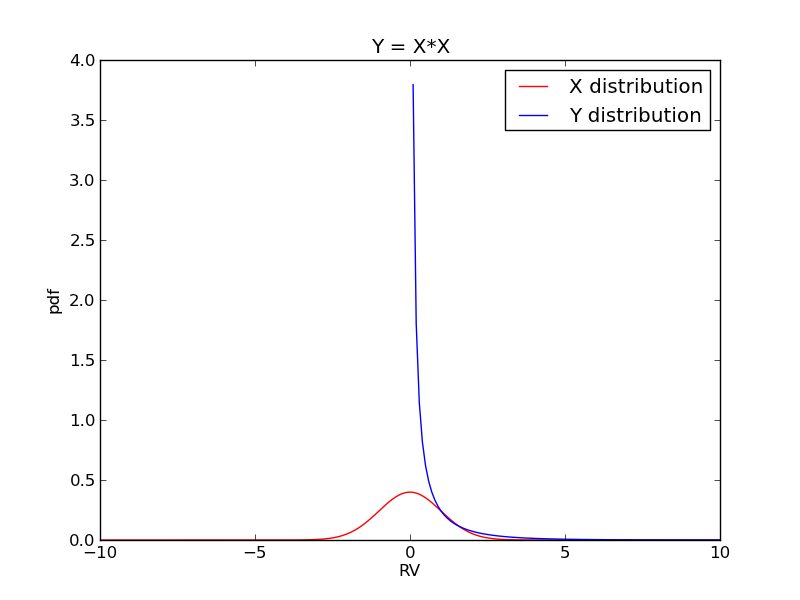
F(y)=P(Y≤y)=P(X2≤y)=P(−y√≤X≤y√)

F(y)=∫y√−y√12π−−√e−x2/2dx=2∫y√012π−−√e−x2/2dx

对上面的F(y)微分，即获得密度函数

f(y)=12π−−√y−1/2e−y/2,0≤y≤∞

绘制密度函数



[复制代码](javascript:void(0);)

import numpy as np

import matplotlib.pyplot as plt

pi = np.pi

x = np.linspace(-10, 10, 200)

y = np.linspace(0.1, 10, 100)

fx = 1/np.sqrt(2\*pi)\*np.exp(-x\*\*2/2)

fy = 1/np.sqrt(2\*pi)\*(y\*\*(-1/2))\*np.exp(-y/2)

plt.plot(x, fx, color = "red", label="X distribution")

plt.plot(y, fy, label="Y distribution")

plt.title("Y = X\*X")

plt.xlabel("RV")

plt.ylabel("pdf")

plt.legend()

plt.show()

[复制代码](javascript:void(0);)

上面的例子展示的是单变量函数，我们看一个多变量函数的例子。即Y=g(X1,X2,...,Xn)，且已知X1,X2,...,Xn的联合分布为f(x1,x2,...,xn)。我们需要找到满足g(x1,x2,...,xn)≤y的区间。

比如，Y=X1+X2，且X1,X2满足如下分布:

f(x1,x2)=12πexp(−12(x21+x22))

为了让x1+x2≤y，我们可以让x1任意取值，而让x2≤y−x1

FY(y)=∫∞−∞∫y−x1−∞f(x1,x2)dx2dx1

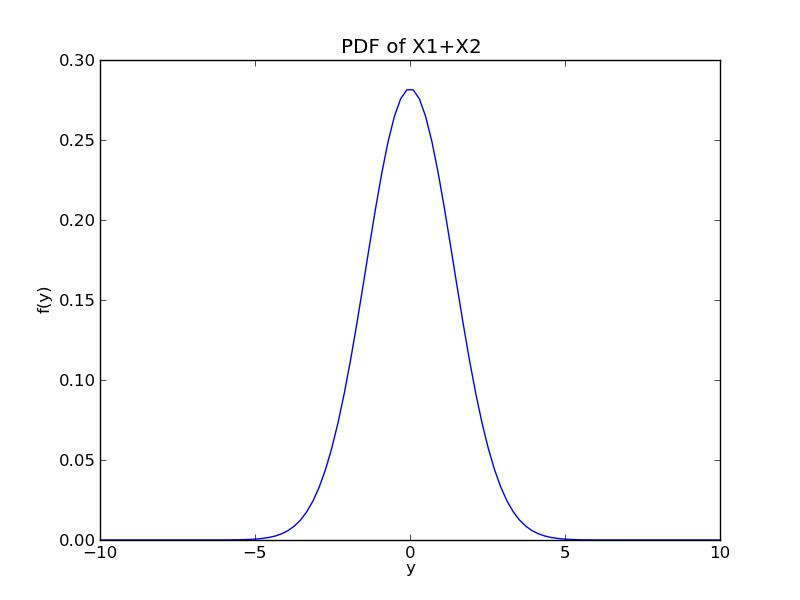
让x\_2 = v - x\_1，有

FY(y)=∫∞−∞∫y−∞f(x1,v−x1)dvdx1=∫y−∞∫∞−∞f(x1,v−x1)dvdx1

微分，可得y的分布为:

fY(y)=∫∞−∞f(x1,y−x1)dx1=∫∞−∞12πexp(−12(x21+(y−x1)2))dx1

上述方程也可以使用数值方法求解:



代码如下:

[复制代码](javascript:void(0);)

# By Vamei

import numpy as np

import scipy.integrate

import matplotlib.pyplot as plt

pi = np.pi

'''

core of the integral

'''

def int\_core(y):

f = lambda x: 1.0/(2\*pi)\*np.exp(-0.5\*(x\*\*2 + (y-x)\*\*2))

return f

'''

calculate f(y)

'''

def density(y):

rlt = scipy.integrate.quad(int\_core(y), -np.inf, np.inf)

return rlt[0]

# get distribution

y = np.linspace(-10, 10, 100)

fy = map(density, y)

plt.plot(y, fy)

plt.title("PDF of X1+X2")

plt.ylabel("f(y)")

plt.xlabel("y")

plt.show()

[复制代码](javascript:void(0);)

上面的int\_core()函数是一个闭包，它表示积分核部分。density()函数用于求某个y值下的积分结果。

(我们也可以利用解析的方法，推导出f(y)满足分布N(0,2√)。如果有微积分基础，可以将此作为练习。)

### 单变量函数的通用公式

上面求新的随机变量分布的步骤较为繁琐。在一些特殊情况下，我们可以直接代入通用公式，来获得新的分布。

(通用公式实际上是从基本方法推导出的数学表达式)

对于单变量函数来说，如果Y=g(X)，g是一个可微并且单调变化的函数 (在该条件，存在反函数g−1，使得X=g−1(Y))。那么我们可以使用下面的通用公式，来获得Y的分布:

fY(y)=fX(g−1(y))⋅ddyg−1(y)

假设X为标准分布，即N(0,1)，且Y=5X+1，那么g−1(y)=(y−1)/5，因此:

fY(y)=fX((y−1)/5)⋅(1/5)=152π−−√e−(y−1)2/(2×25)

可以看到，新的分布是一个μ=1,σ=5的正态分布，即N(1,5)

并不是所有的函数都有反变换，所以这里的“通用”公式并不能适用于所有的情况。

### 多变量函数的通用公式

在一些特殊情况下，我们可以使用多变量函数的通用公式。

如果U=g1(X,Y),V=g2(X,Y)，且存在反变换，使得

X=h1(U,V)

Y=h2(U,V)

那么，我们可以通过如下公式，从X,Y的分布获得U,V的联合分布:

fUV(u,v)=fXY(h1(u,v),h2(u,v))|J|

J表示雅可比变换(Jacobian tranformation)，表示如下

J=∣∣∣∣∂x∂u∂y∂u∂x∂v∂y∂v∣∣∣∣=∂x∂u∂y∂v−∂x∂v∂y∂u

如果X和Y是独立的随机变量，且有相同的分布

f(x)=e−x,x≥0

。如果U=X+Y,V=Y，求U和V的联合分布。

由于X和Y独立，所以

fXY(x,y)=f(x)f(y)=e−xe−y

根据U=X+Y，V=Y，可以得到u≥0,v≥0, 且有:

X=U−V

Y=V

因此

f(u,v)=e−(u−v)e−v=e−u,u≥0,v≥0

### 总结

通过随机变量的函数，我们可以利用已知随机变量，创建新的随机变量，并获得其分布。

# [概率论09 期望](http://www.cnblogs.com/vamei/p/3230753.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

### 描述量

描述随机变量最完备的方法是写出该随机变量的概率分布。然而，正如我们在前面章节看到的，概率分布的表达往往都比较复杂，信息量很大。这如同我们购置汽车的时候，一辆汽车的全面数据可以说是海量的，比如汽车尺寸，油箱大小等等。我们选择一辆汽车时，往往只使用有限的几个具有代表性的量来代表汽车的主要特征，比如排气量，最大马力。我们信赖这几个量，因为它们可以“粗糙”的描述汽车的主要性能。这些量是汽车全面数据的一个缩影。

类似的，统计学家也设计了这样的投影系统，将全面的概率分布信息量投射到某几个量上，来代表随机变量的主要特征，从而掌握该随机变量的主要“性能”。这样的一些量称为随机变量的描述量(descriptor)。比如期望用于表示分布的中心位置，方差用于表示分布的分散程度等等。这些描述量可以迅速的传递其概率分布的一些主要信息，允许我们在深入研究之前，先对其特征有一个大概了解。

(买西瓜之前，先听听声音，可以对西瓜的成熟度有个了解。)

### 期望

期望(expectation)是概率分布的一个经典描述量，它有很深的现实根源。在生活中，我们往往对未知事件有一个预期，也就是我们的期望。比如，我们会根据自己的平时成绩，来期望高考分数。现实生活中的期望可以是许多因素的混合，比如历史表现和主观因素。



你的期望是什么？

在概率论中，我们更加定量的对未知结果进行预估。根据概率分布，我们以概率值为权重，加权平均所有可能的取值，来获得了该随机变量的期望(expectation):

E(x)=∑ixip(xi)

如果某个取值概率较大，那么它就在最终结果中占据较大的分量。期望是一个非常简单而直观的概念。期望常用字母μ表示 (μ同样是高斯分布的一个参数，我们将马上看到，为什么同一个字母用在两个地方)。

期望在生活中非常常见，特别在估计收益的时候。比如，买一张彩票的收益为一个随机变量X。该彩票售价为2元，有三位数，每位数可以从0到9中任意选择。每期有一个随机选择的号获奖，奖金1000元。那么，X的分布为:

p(−2)=999/1000

p(998)=1/1000

因此，

E(X)=−2×p(−2)+998×p(998)=−1.0

也就是说，如果买一张彩票，收益的期望为损失1元。

期望是在事件还没确定时，根据概率，对平均结果的估计。如果事件发生，结果并不是期望值。但是，如果重复进行大量实验，其结果的平均值会趋近期望值。需要注意的是，我们将期望写成E(X)，这表示的是一个数值，而不是一个随机变量的函数。

基于相似的道理，可以用下面的积分公式，计算连续随机变量的期望：

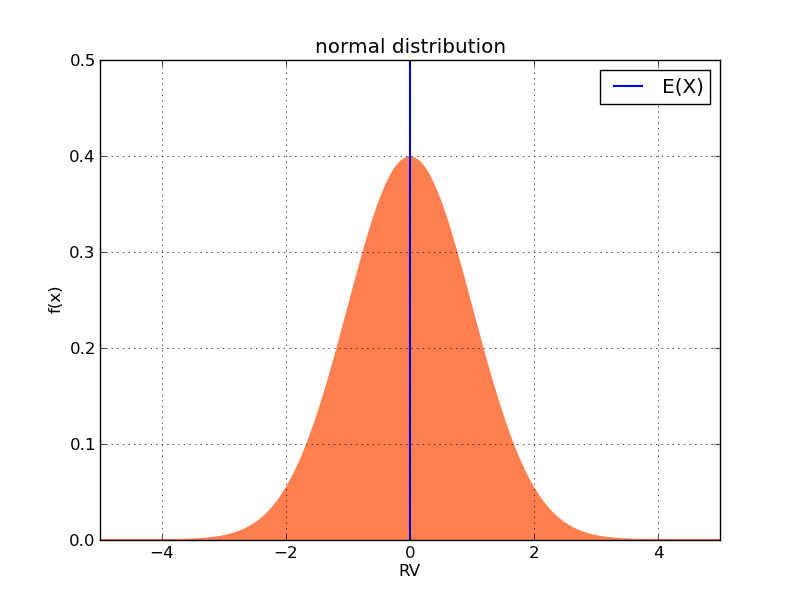
E(X)=∫+∞−∞xf(x)dx

**正态分布的期望**

E(X)=1σ2π−−√∫+∞−∞xe−(x−μ)2/2σ2dx=μ

即，分布的参数μ就是正态分布的期望！这也是μ常用于表示期望的原因。

回忆正态分布的密度函数曲线，x=μ是分布曲线的对称轴。如果将密度函数曲线下的面积看做一个“物品”，那么x=μ是该“物品”的重心所在。比如μ=0,σ=1时，



代码如下:

[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import norm

import numpy as np

import matplotlib.pyplot as plt

rv = norm(loc=0, scale = 1)

x = np.linspace(-5, 5, 200)

plt.fill\_between(x, rv.pdf(x), y2=0.0 color="coral", label="N(0,1)")

plt.axvline(x = rv.mean(), label="E(X)", linewidth=1.5, color="blue")

plt.legend()

plt.grid(True)

plt.xlim([-5, 5])

plt.ylim([-0.0, 0.5])

plt.title("normal distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

上面的代码中，rv是一个随机变量对象，调用mean()方法，可以计算该随机变量的期望值。

**指数分布的期望**

根据指数分布的表达式，

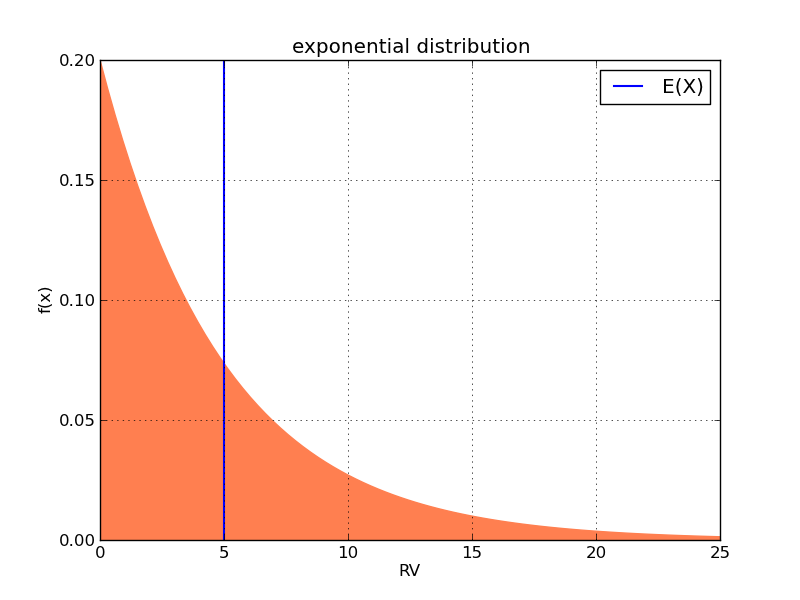
f(x)={λe−λx0ififx≥0x<0

它的期望为:

E(x)=1/λ

对于λ=0.2的指数分布，它的期望值为5。

可以通过编程，来计算指数分布的期望。如下图所示:



[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import expon

import numpy as np

import matplotlib.pyplot as plt

rv = expon(scale = 5)

x = np.linspace(0.0, 30, 100)

print rv.pdf(x)

plt.fill\_between(x, rv.pdf(x), y2=0, color="coral", label="0.2")

plt.axvline(x = rv.mean(), label="E(X)", linewidth=1.5, color="blue")

plt.grid(True)

plt.legend()

plt.xlim([0, 25])

plt.ylim([0, 0.2])

plt.title("exponential distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

### 期望的性质

期望有一些很有用的性质：

**性质1.**

如果Y=g(X)，那么当X为离散随即变量，且∑|g(x)|p(x)<∞ (该条件保证下面的累加为有限值)

E(Y)=∑xg(x)p(x)

当X为连续随机变量，且∫|g(x)|f(x)dx<∞ (该条件保证下面的积分为有限值)

E(Y)=∫∞−∞g(x)f(x)dx

回忆随机变量的函数。X和Y之间存在对应关系。Y的概率分布，等于对应X的概率分布。因此，Y=g(X)根据X概率的加权平均，就是Y的期望。

**性质2.**

Y=g(X1,X2,...,Xn)。如果Xi是离散的，且有分布p(x1,...,xn)，那么当∑x1,...,xn|g(x1,...,xn)|p(x1,...,xn)<∞时，有

E(Y)=∑x1,...,xng(x1,...,xn)p(x1,...,xn)

如果Xi是连续的，且有分布f(x1,...,xn)，那么当∫∫...∫|g(x1,...,xn)|f(x1,...,xn)dx<∞时，有

E(Y)=∫∫...∫g(x1,...,xn)f(x1,...,xn)dx

这一性质与上面一个性质类似，只不过换成多变量联合分布的情况。

利用性质1和性质2，我们可以根据原随机变量的分布，计算随机变量函数的期望值。

**性质3.**

如果X和Y是独立随机变量，而g和h是两个函数，如果E[g(X)],E[h(Y)]存在，那么有

E[g(X)h(Y)]=E[g(X)]E[h(Y)]

根据独立随机变量的性质，我们可以将联合分布写成f(x)和f(y)的乘积。结合性质2，即可得出上面的结论。

一个特别的情况是，如果X和Y独立，那么E(XY)=E(X)E(Y)。

(即g(X)=X,h(Y)=Y的情况)

**性质4.**

如果Y=a+∑ni=1biXi，而Xi的期望为E(Xi)，那么

E(Y)=a+∑i=1nbiE(Xi)

这说明，期望是一个线性运算。随机变量线性组合的期望，等于期望的线性组合。

我们可以假设f(x\_i, ..., x\_n)的联合分布，并根据性质2来证明性质4。对联合分布的积分，可以得到单随机变量的边缘分布，从而获得单随机变量的期望。

上面四个性质的一个主要功能是，利用已知的期望值，来计算未知的期望值。有些随机变量的期望值比较难以通过定义计算。利用上面的性质，进行合理的变化，更容易计算其期望。

比如，计算二项分布的期望。二项分布是进行n次实验，其中成功的次数Y。每次成功的概率为p。根据定义计算

E(Y)=∑k=0n(nk)kpk(1−p)n−k

上面的计算并不容易。另一方面，观察可知，每次试验成功的次数X是伯努利分布，即

p(1)=p,p(0)=1−p

E(X)=p

二项分布Y可以表示为n个伯努利分布的和，即

Y=∑k=inXi

所以

E(Y)=∑k=inE(Xi)=np

### 条件期望

条件期望将期望用于条件概率。我们已经知道，条件概率是事件B条件下， A的概率，即P(A|B)。条件概率只不过是在一个缩小了的样本空间B上，重新计算A的概率。条件概率的A与B可以是随机变量，比如P(X|Y=y)，即“随机变量Y等于y”是条件，在该条件下，随机变量X的随机分布。

(在连续随机变量的情况下，我们使用条件密度函数f(x|Y=y)来描述条件分布)  
  
对于一个已知的分布，我们可以求得条件分布的期望。对于离散随机变量:

E(X|Y=y)=∑ixip(xi|Y=y)

其中，xi为该离散随机变量的可能取值。也就是，在一个新的样本空间(Y = y)上，随机变量X的期望值。  
  
对于连续随机变量，其条件期望为:

E(X|Y=y)=∫+∞−∞xf(x|Y=y)dx

一个随机变量的期望为一个数值。但一个条件分布的期望，比如E(X|Y=y)，会随着随机变量Y的变化而变化。所以，条件期望是随机变量Y的函数。根据随机变量的函数的概念，E(X|Y=y)可以看作一个新的随机变量。我们可以进一步得到这一新的随机变量的期望E(E(X|Y))。

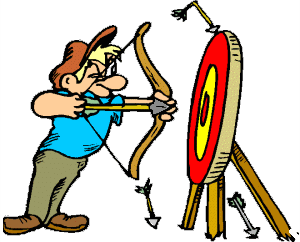
### 总结

期望是随机变量分布的一个描述量，用“概率加权平均”来计算，表达随机变量的预期。

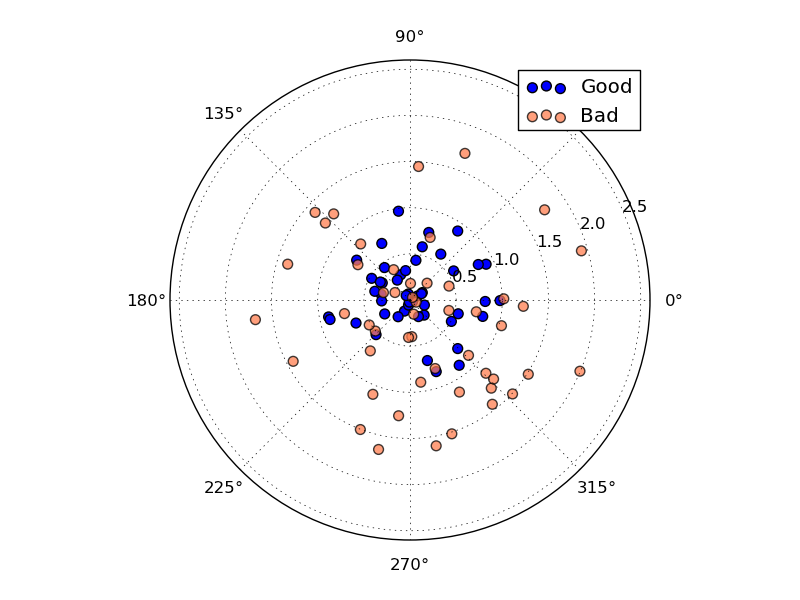
# [概率论10 方差与标准差](http://www.cnblogs.com/vamei/p/3232313.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

除了[期望](http://www.cnblogs.com/vamei/p/3230753.html)，方差(variance)是另一个常见的分布描述量。如果说期望表示的是分布的中心位置，那么方差就是分布的离散程度。方差越大，说明随机变量取值越离散。



比如射箭时，一个优秀的选手能保持自己的弓箭集中于目标点附近，而一个经验不足的选手，他弓箭的落点会更容易散落许多地方。



上面的靶上有两套落点。尽管两套落点的平均中心位置都在原点 (即期望相同），但两套落点的离散程度明显有区别。蓝色的点离散程度更小。

数学上，我们用方差来代表一组数据或者某个概率分布的离散程度。可见，方差是独立于期望的另一个对分布的度量。两个分布，完全可能有相同的期望，而方差不同，正如我们上面的箭靶。

### 方差

对于一个随机变量X来说，它的方差为:

Var(X)=E[(X−μ)2]

其中，μ表示X的期望值，即μ=E(X)。

我们可以代入[期望](http://www.cnblogs.com/vamei/p/3230753.html)的数学表达形式。比如连续随机变量：

Var(X)=E[(X−μ)2]=∫+∞−∞(x−μ)2f(x)dx

方差概念背后的逻辑很简单。一个取值与期望值的“距离”用两者差的平方表示。该平方值表示取值与分布中心的偏差程度。平方的最小取值为0。当取值与期望值相同时，此时不离散，平方为0，即“距离”最小；当随机变量偏离期望值时，平方增大。由于取值是随机的，不同取值的概率不同，我们根据概率对该平方进行加权平均，也就获得整体的离散程度——方差。

方差的平方根称为标准差(standard deviation, 简写std)。我们常用σ表示标准差

σ=Var(X)−−−−−−√

标准差也表示分布的离散程度。

**正态分布的方差**

根据上面的定义，可以算出正态分布

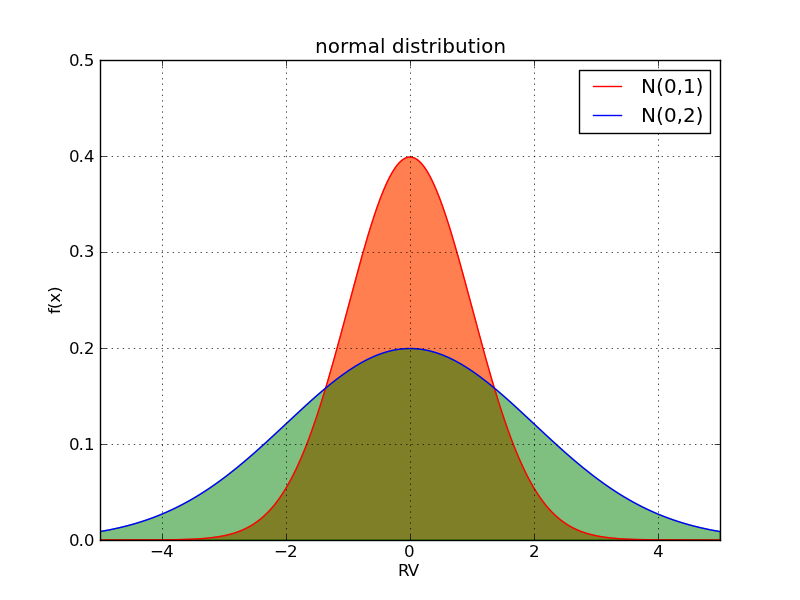
E(X)=1σ2π−−√∫+∞−∞xe−(x−μ)2/2σ2dx

的方差为

Var(X)=σ2

正态分布的标准差正等于正态分布中的参数σ。这正是我们使用字母σ来表示标准差的原因！

可以预期到，正态分布的σ越大，分布离散越大，正如我们从下面的分布曲线中看到的:

当方差小时，曲线下的面积更加集中于期望值0附近。当方差大时，随机变量更加离散。此时分布曲线的“尾部”很厚，即使在取值很偏离0时，比如x=4时，依然有很大的概率可以取到。

代码如下:

[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import norm

import numpy as np

import matplotlib.pyplot as plt

# Note the difference in "scale", which is std

rv1 = norm(loc=0, scale = 1)

rv2 = norm(loc=0, scale = 2)

x = np.linspace(-5, 5, 200)

plt.fill\_between(x, rv1.pdf(x), y2=0.0, color="coral")

plt.fill\_between(x, rv2.pdf(x), y2=0.0, color="green", alpha = 0.5)

plt.plot(x, rv1.pdf(x), color="red", label="N(0,1)")

plt.plot(x, rv2.pdf(x), color="blue", label="N(0,2)")

plt.legend()

plt.grid(True)

plt.xlim([-5, 5])

plt.ylim([-0.0, 0.5])

plt.title("normal distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

**指数分布的方差**

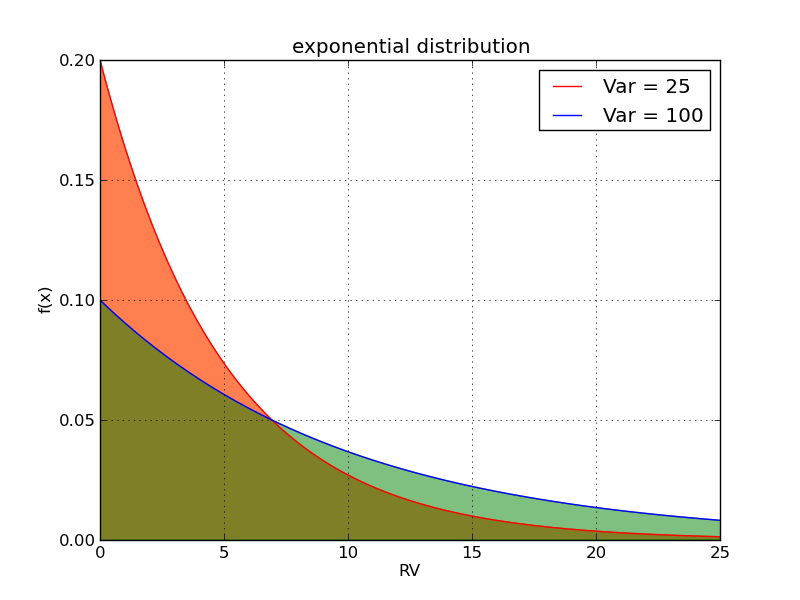
指数分布的表达式为

f(x)={λe−λx0ififx≥0x<0

它的方差为

Var(X)=1λ2

如下图所示:

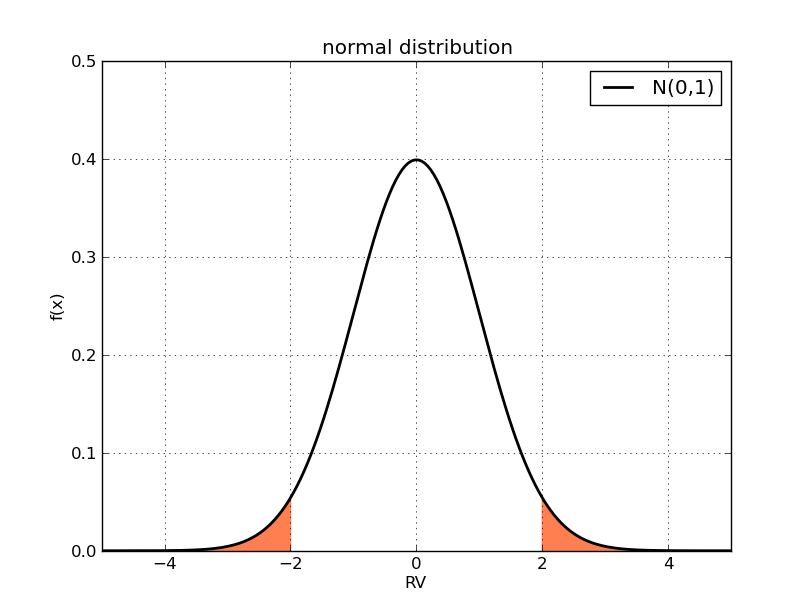


### Chebyshev不等式

我们一直在强调，标准差(和方差)表示分布的离散程度。标准差越大，随机变量取值偏离平均值的可能性越大。如何定量的说明这一点呢？我们可以计算一个随机变量与期望偏离超过某个量的可能性。比如偏离超过2个标准差的可能性。即

P(|X−μ|>2σ)

这个概率依赖于分布本身的类型。比如正态分布N(0,1)，这一概率即为x大于2，或者x小于-2的部分对应的曲线下面积：



实际上，无论μ和σ如何取值，对于正态分布来说，偏离期望超过两个标准差的概率都相同，约等于0.0455 (可以根据正态分布的表达式计算)。随机变量的取值有约95.545%的可能性落在正负两个标准差的区间内，即从-2到2。如果我们放大区间，比如正负三个标准差，这一概率超过99%。我们可以相当有把握的说，随机变量会落正负三个标准差之内。上面的论述并不依赖于标准差的具体值。这里可以看到标准差所衡量的“离散”的真正含义：如果取相同概率的极端值区间，比如上面的0.0455，标准差越大，该极端值区间距离中心值越远。

然而，上面的计算和表述依赖于分布的类型（正态分布）。如何将相似的方差含义套用在其它随机变量身上呢？

Chebyshev不等式让我们摆脱了对分布类型的依赖。它的叙述如下：

对于任意随机变量X，如果它的期望为μ，方差为σ2，那么对于任意t>0，

P(|X−μ|>t)≤σ2t

无论X是什么分布，上述不等式成立。我们让t=2σ，那么

P(|X−μ|>2σ)≤0.25

也就是说，X的取值超过两个正负标准差的可能性最多为25%。换句话说，随机变量至少有75%的概率落在正负两个标准差的范围内。（显然这是最“坏”的情况下。正态分布显然不是”最坏“的）

绘图代码如下

[复制代码](javascript:void(0);)

from scipy.stats import norm

import numpy as np

import matplotlib.pyplot as plt

# Note the difference in "scale", which is std

rv1 = norm(loc=0, scale = 1)

x1 = np.linspace(-5, -1, 100)

x2 = np.linspace(1, 5, 100)

x = np.linspace(-5, 5, 200)

plt.fill\_between(x1, rv1.pdf(x1), y2=0.0, color="coral")

plt.fill\_between(x2, rv1.pdf(x2), y2=0.0, color="coral")

plt.plot(x, rv1.pdf(x), color="black", linewidth=2.0, label="N(0,1)")

plt.legend()

plt.grid(True)

plt.xlim([-5, 5])

plt.ylim([-0.0, 0.5])

plt.title("normal distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

### 总结

我们引入了一个新的分布描述量：方差。它用于表示分布的离散程度。

标准差为方差的平方根。

方差越大，“极端区间”偏离中心越远。

# [概率论11 协方差与相关系数](http://www.cnblogs.com/vamei/p/3416138.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

前面介绍的分布描述量，比如期望和方差，都是基于单一随机变量的。现在考虑多个随机变量的情况。我们使用联合分布来表示定义在同一个样本空间的多个随机变量的概率分布。

联合分布中包含了相当丰富的信息。比如从联合分布中抽取某个随机变量的边缘分布，即获得该随机变量的分布，并可以据此，获得该随机变量的期望和方差。这样做是将视线限制在单一的一个随机变量上，我们损失了联合分布中包含的其他有用信息，比如不同随机变量之间的互动关系。为了了解不同随机变量之间的关系，需要求助其它的一些描述量。

### 协方差

协方差(covariance)表达了两个随机变量的协同变化关系。我们取一个样本空间，即学生的体检数据。学生的身高为随机变量X，学生的体重为随机变量Y。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 160cm | 170cm | 180cm |
| 60kg | 0.2 | 0.05 | 0.05 |
| 70kg | 0.05 | 0.3 | 0.05 |
| 80kg | 0.05 | 0.05 | 0.2 |

根据上表，大的身高(180cm)和大的体重(80kg)同时出现的概率较大(0.2)，小的身高值(160cm)和小的体重(60kg)的概率也较大(0.2)。偏大的身高往往伴随偏大的体重，偏小的身高常伴随偏小的体重。这种“大”伴随着“大”，“小”伴随着“小”的情形，叫做正相关。根据上面的数据，身高和体重两个随机变量正相关性很强。

另一方面，如果“大”配“小”，“小”配“大”的概率很高，那么两个随机变量负相关。“最萌身高差”是负相关的一个范例。（样本空间为情侣的身高信息。可以定义男生身高为一个随机变量，女生身高为另一个随机变量）



正如其他的分布描述量一样，协方差从概率分布中提取信息，让我们获知分布的“性能”。对于一个已知的联合分布来说，任意两个随机变量之间都可以计算出一个协方差，即一个数值。

### 定义

协方差的定义如下，如果X和Y是联合分布的随机变量，且分别有期望μX，μY，那么X和Y的协方差为

Cov(X,Y)=E[(X−μX)(Y−μY)]

协方差的定义基于期望。根据期望的定义，协方差可以直接用于离散随机变量和连续随机变量。

我们已经知道，期望是某个随机变量根据概率的加权平均。我们所要加权平均的目标是X−μX和Y−μY的乘积。随机变量和期望的差，代表了随机变量的取值和中心值的偏离程度，也就是我们上面所谓的“偏大”或者“偏小”的情况：正值的偏离表示“偏大”，负值的偏离表示“偏小”。如果是正相关，即大配大，小配小的情况，那么这一乘积为正；如果是负相关，乘积为负。所以，通过(X−μX)(Y−μY)这个量，我们表达了X和Y的相关性。

回到刚才的数据来计算相关性，

|  |  |  |  |
| --- | --- | --- | --- |
|  | 160cm | 170cm | 180cm |
| 60kg | 0.2 | 0.05 | 0.05 |
| 70kg | 0.05 | 0.3 | 0.05 |
| 80kg | 0.05 | 0.05 | 0.2 |

让身高为X，体重为Y。我们可以通过边缘分布，来分别获得X和Y的分布(回忆一下)。求得X和Y的期望，分别为170和70。计算各个格子中的(X−μX)(Y−μY)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 160cm | 170cm | 180cm |
| 60kg | 100 | 0 | -100 |
| 70kg | 0 | 0 | 0 |
| 80kg | -100 | 0 | 100 |

上面的两个表，对应的格子相乘，并求和，就得到协方差:

Cov(X,Y)=0.2×100+0.2×100+0.05×(−100)+0.05×(−100)=30

在上面的计算中，正相关的项目都分配有比较大的概率值。最终的协方差也是一个正值。

根据期望的性质，我们可以改写协方差的表达形式:

Cov(X,Y)=E(XY−XμX−YμX+μXμY)=E(XY)−E(X)μX−E(Y)μY+μXμY=E(XY)−E(X)E(Y)

当X和Y独立时，有E(XY)=E(X)E(Y)，Cov(X,Y)=0。

(注意，Cov(X,Y)=0并不意味着X和Y独立)

### 相关系数

正的协方差表达了正相关性，负的协方差表达了负相关性。对于同样的两个随机变量来说，计算出的协方差越大，相关性越强。

但随后一个问题，身高和体重的协方差为30，这究竟是多大的一个量呢？如果我们又发现，身高与鞋号的协方差为5，是否说明，相对于鞋号，身高与体重的的相关性更强呢？

这样横向对比超出了协方差的能力范围。从日常生活经验来说，体重的上下浮动大约为20kg，而鞋号的上下浮动大约可能只是5个号码。所以，对于体重来说，5kg与中心的偏离并不算大，而5个号码的鞋号差距，就可能是最极端的情况了。假设身高和体重的相关强度，与身高和鞋码的相关强度类似，但由于体重本身的数值上下浮动更大，所计算出的协方差也会更大。另一个情况，依然是计算身高与体重的协方差。数据完全不变，而只更改单位。我们的体重用克而不是千克做单位，计算出的协防差是原来数值的1000倍！

为了能进行这样的横向对比，我们需要排除用统一的方式来定量某个随机变量的上下浮动。这时，我们计算相关系数(correlation coefficient)。相关系数是“归一化”的协方差。它的定义如下:

ρ=Cov(X,Y)Var(X)Var(Y)−−−−−−−−−−−−√

相关系数是用协方差除以两个随机变量的标准差。相关系数的大小在-1和1之间变化。再也不会出现因为计量单位变化，而数值暴涨的情况了。

依然使用上面的身高和体重数据，可以计算出

Var(X)=0.3×(60−70)2+0.3×(80−70)2=60

Var(Y)=0.3×(180−170)2+0.3×(160−170)2=60

ρ=30/60=0.5

这样一个“归一化”了的相关系数，更容易让人把握到相关性的强弱，也更容易在不同随机变量之间，做相关性的横向比较。

### 双变量正态分布

双变量正态分布是一种常见的联合分布。它描述了两个随机变量X1和X2的概率分布。概率密度的表达式如下：

f(x1,x2)=12πσ1σ21−ρ2−−−−−√exp[−z2(1−ρ2)]

其中，

z=(x1−μ1)2σ21−2ρ(x1−μ1)(x2−μ2)σ1σ2+(x2−μ2)2σ22

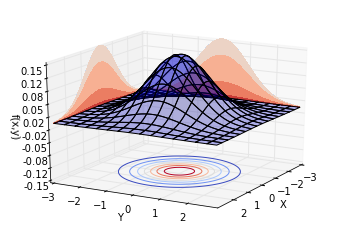
X1和X2的边缘密度分别为两个正态分布，即正态分布N(μ1,σ1), N(μ2,σ2)。

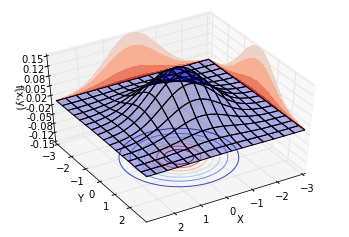
另一方面，除非ρ=0，否则联合分布也并不是两个正态分布的简单相乘。可以证明，ρ正是双变量正态分布中，两个变量的相关系数。

我们现在绘制该分布的图像。可惜的是，现在的scipy.stats并没有该分布。我们需要自行编写。

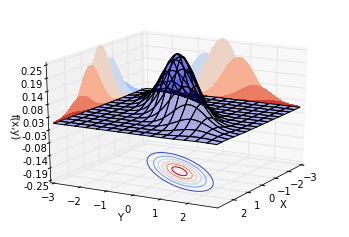
选取所要绘制的正态分布，为了简单起见，让μ1=0, μ2=0, σ1=1,σ2=1。

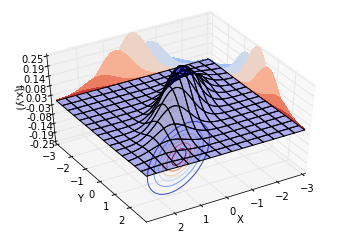
我们先让ρ=0，此时的联合分布相当于两个正态分布的乘积。绘制不同视角的同一分布，结果如下。可以看到，概率分布是中心对称的。





再让ρ=0.8，也就是说，两个随机变量的相关系数为0.8。绘制不同视角的同一分布，结果如下。可以看到，概率分布并不中心对称。沿着Y=X这条线，概率曲面隆起，概率明显比较高。而沿着Y=−X这条线，概率较低。这也就是我们所说的正相关。





现在，ρ对于我们来说，有了更具体的现实意义。:-)

[复制代码](javascript:void(0);)

# By Vamei

from scipy.stats import norm

import numpy as np

# this function is to generate a pdf of bivariate normal distribution

def bivar\_norm(mu1, mu2, sigma1, sigma2, rho):

# pdf of bivariate norm

def pdf(x1, x2):

# get z

part1 = (x1 - mu1)\*\*2/sigma1\*\*2

part2 = - 2.\*rho\*(x1 - mu1)\*(x2 - mu2)/sigma1\*sigma2

part3 = (x2 - mu2)\*\*2/sigma2\*\*2

z = part1 + part2 + part3

cof = 1./(2.\*np.pi\*sigma1\*sigma2\*np.sqrt(1 - rho\*\*2))

return cof\*np.exp(-z/(2.\*(1 - rho\*\*2)))

return pdf

pdf1 = bivar\_norm(0, 0, 1, 1, 0)

pdf2 = bivar\_norm(0, 0, 1, 1, 0.8)

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib import cm

from matplotlib.ticker import LinearLocator, FormatStrFormatter

import matplotlib.pyplot as plt

# plot function

def space\_surface(pdf, xp, yp, zlim, rot1=30, rot2=30):

fig = plt.figure()

ax = fig.gca(projection='3d')

X = np.arange(\*xp)

Y = np.arange(\*yp)

X, Y = np.meshgrid(X, Y)

Z = pdf(X, Y)

surf = ax.plot\_surface(X, Y, Z, rstride=8, cstride=8,

alpha = 0.3)

cset = ax.contour(X, Y, Z, zdir='z', offset=zlim[0], cmap=cm.coolwarm)

cset = ax.contourf(X, Y, Z, zdir='x', offset=xp[0], cmap=cm.coolwarm)

cset = ax.contourf(X, Y, Z, zdir='y', offset=yp[0], cmap=cm.coolwarm)

for angle in range(rot1 + 0, rot1 + 360):

ax.view\_init(rot2, angle)

ax.set\_zlim(\*zlim)

ax.zaxis.set\_major\_locator(LinearLocator(10))

ax.zaxis.set\_major\_formatter(FormatStrFormatter('%.02f'))

ax.set\_xlabel("X")

ax.set\_ylabel("Y")

ax.set\_zlabel("f(x,y)")

# fig.colorbar(surf, shrink=0.5, aspect=5)

xp = [-3, 3, 0.05]

yp = [-3, 3, 0.05]

zlim1 = [-0.15, 0.15]

zlim2 = [-0.25, 0.25]

space\_surface(pdf1, xp, yp, zlim1, 30, 20)

space\_surface(pdf1, xp, yp, zlim1, 60, 45)

space\_surface(pdf2, xp, yp, zlim2, 30, 20)

space\_surface(pdf2, xp, yp, zlim2, 60, 45)

[复制代码](javascript:void(0);)

### 总结

协方差

“归一化”的度量: 相关系数

# [概率论12 矩与矩生成函数](http://www.cnblogs.com/vamei/p/3418589.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

我们重新回到对单随机变量分布的研究。描述量是从分布中提取出的一个数值，用来表示分布的某个特征。之前使用了两个描述量，即期望和方差。在期望和方差之外，还有其它的描述量吗？

### 斜度

值得思考的是，期望和方差足以用来描述一个分布吗？如果答案是可以，那么我们就没有必要寻找其它描述量的。事实上，这两个描述量并不足以完整的描述一个分布。

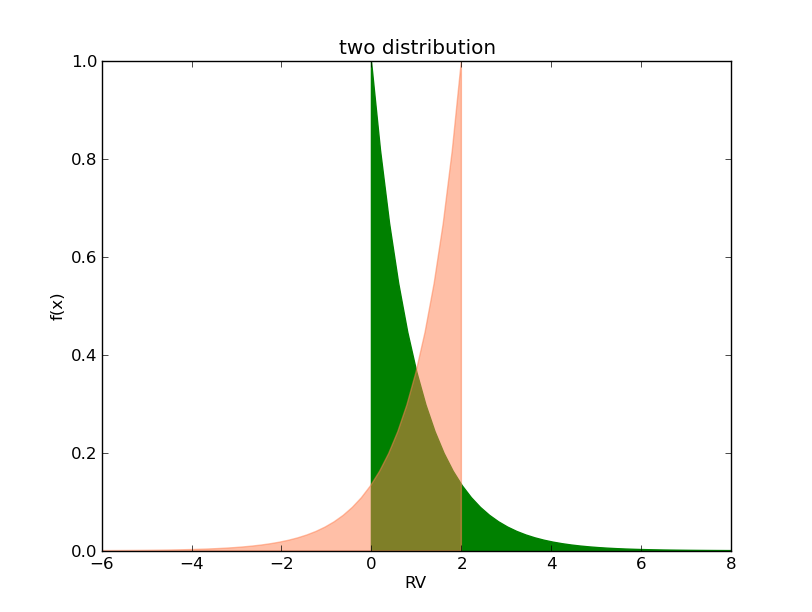
我们来看两个分布，一个是指数分布:

f(x)={ex0ififx≥0x<0

它的期望为E(x)=1，方差为Var(x)=1。  
  
我们用Y = 2-X来获得一个新的随机变量，及其分布:

f(y)={e2−y0ifify≤2y>2

该密度曲线与原来的密度曲线关于直线X=1对称，与原来的分布有相同的期望值和方差。期望为E(x)=1，方差为Var(x)=1

我们绘制两个分布的密度曲线，如下图:  
  
  
可以看到，即使期望值和方差保持不变，两个分布曲线明显不同。第一条曲线下的面积偏向左，而第二条曲线则向右侧倾斜。为了表达分布的这一特征，我们引入一个新的描述量，斜度(skewness)。它的定义如下:

Skew(X)=E[(X−μ)3]

上面两个分布，第一条曲线向左偏斜，斜度分别为2。另一条曲线的斜度为-2。很明显，斜度的不同可以带来差别巨大的分布(即使期望和方差都相同)。

绘制程序如下

[复制代码](javascript:void(0);)

from scipy.stats import expon

import numpy as np

import matplotlib.pyplot as plt

rv = expon(scale = 1)

x1 = np.linspace(0, 20, 100)

x2 = np.linspace(-18, 2, 100)

y1 = rv.pdf(x1)

y2 = rv.pdf(2 - x2)

plt.fill\_between(x1, y1, 0.0, color = "green")

plt.fill\_between(x2, y2, 0.0, color = "coral", alpha = 0.5)

plt.xlim([-6, 8])

plt.title("two distribution")

plt.xlabel("RV")

plt.ylabel("f(x)")

plt.show()

[复制代码](javascript:void(0);)

### 矩

观察方差和斜度的定义，

Var(X)=E[(X−μ)2]

Skew(X)=E[(X−μ)3]

都是X的函数的期望。它们的区别只在于函数的形式，即(X−μ)的乘方次数不同。方差为2次方，斜度为3次方。  
  
上面的描述量都可以归为“矩”(moment)的一族描述量。类似于方差和斜度这样的，它们都是(X−μ)乘方的期望，称为中心矩(central moment)。E[(x−μ)k]称为k阶中心矩，表示为μk，其中k = 2, 3, 4, ...

还有另一种是原点矩(moment about the origin)，是X乘方的期望。 E[Xk]称为k阶原点矩，表示为μ′k，其中k = 1, 2, 3, ...

期望是一阶原点矩:

E(X)=E(X1)

### 矩生成函数

除了表示中心、离散程序、斜度这些特性外，更高阶的矩可以描述分布的其它特性。矩统计中有重要的地位，比如参数估计的一种重要方法就是利用了矩。然而，根据矩的定义，我们需要对不同阶的X幂求期望，这个过程包含复杂的积分过程，并不容易。矩同样催生了矩生成函数(moment generating function)，它是求解矩的一样有力武器。

在了解矩生成函数之前，先来回顾幂级数(power series)。幂级数是不同阶数的乘方(比如1,x,x2,x3...)的加权总和：

∑i=1+∞aixi

ai是一个常数。

幂级数是数学中的重要工具，它的美妙之处在于，解析函数都可以写成幂级数的形式，比如三角函数sin(x)可以写成：

sin(x)=x−x33!+x55!−x77!+...

将解析函数分解为幂级数的过程，就是泰勒分解(Taylor）。我们不再深入其具体过程。xn是很简单的一种函数形式，它可以无限次求导，求导也很容易。这一特性让幂级数变得很容易处理。将解析函数写成幂级数，就起到化繁为简的效果。

(幂级数这一工具在数学上的用途极其广泛，它用于数学分析、微分方程、复变函数…… 不能不说，数学家很会活用一种研究透了的工具)

如果我们将幂级数的x看作随机变量X，并求期望。根据期望可以线性相加的特征，有:

E(f(X))=a0+a1E(X)+a2E(X2)+a3E(X3)+...

我们可以通过矩，来计算f(X)的期望。

另一方面，我们可否通过解析函数来获得矩呢？我们观察下面一个指数函数，写成幂级数的形式：

etx=1+tx+(tx)22!+(tx)33!+(tx)44!...

我们再次将x看作随机变量X，并对两侧求期望，即

E(etX)=1+tE(X)+t2E(X2)2!+t3E(X3)3!+t4E(X4)4!...

即使随机变量的分布确定，E(etX)的值还是会随t的变化而变化，因此这是一个关于t的函数。我们将它记为M(t)，这就是矩生成函数(moment generating function)。对M(t)的级数形式求导，并让t等于0，可以让高阶的t的乘方消失，只留下E(X)，即

M′(0)=E(X)

即一阶矩。如果继续求高阶导，并让t等于0，可以获得高阶的矩。

M(r)(0)=E(Xr)

有趣的是，多次求导系数正好等于幂级数系数中的阶乘，所以可以得到上面优美的形式。我们通过幂级数的形式证明了，对矩生成函数求导，可以获得各阶的矩。相对于积分，求导是一个容易进行的操作。

### 矩生成函数的性质

矩生成函数的一面是幂级数，我们已经说了很多。矩生成函数的另一面，是它的指数函数的解析形式。即

M(t)=E[etX]=∫∞−∞etxf(x)dx

在我们获知了f(x)的具体形式之后，我们可以利用该积分获得矩生成函数，然后求得各阶的矩。当然，你也可以通过矩的定义来求矩。但许多情况下，上面指数形式的积分可以使用一些已有的结果，所以很容易获得矩生成函数。矩生成函数的求解矩的方式会便利许多。

矩生成函数的这一定义基于期望，因此可以使用期望的一些性质，产生有趣的结果。

性质1 如果X的矩生成函数为$MX(t)]，且[$Y=aX+b，那么

MY(t)=eatMX(bt)

(将Y写成指数形式的期望，很容易证明该结论)

性质2 如果X和Y是独立随机变量，分别有矩生成函数MX,MY。那么对于随机变量Z=X+Y，有

MZ(t)=MX(t)MY(t)

(基于独立随机变量乘积的期望，等于随机变量期望的乘积)

练习:

推导Poisson分布的矩生成函数

### 总结

矩

矩生成函数

# [概率论13 中心极限定律](http://www.cnblogs.com/vamei/p/3460965.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

在整个概率论中，核心的问题是随机变量的分布。正如我们在[离散分布](http://www.cnblogs.com/vamei/p/3198371.html)和[连续分布](http://www.cnblogs.com/vamei/p/3199522.html)中看到的，分布有许多种类。更夸张的是，在满足[概率公理](http://www.cnblogs.com/vamei/p/3194360.html)的前提下，我们完全可以自行设计分布。想像一下，如果有一天数学书上印一个Vamei分布，这是多么美好的事情啊！然而，这一愿望并不那么容易实现。那些“名流”分布，比如“泊松”，“高斯”，“伯努利”分布，往往在理论上很重要，所以得到了数学家的深入研究。“知名”分布的特性(比如它们的期望、方差、累计概率函数)可以很容易在数学手册中找到，这些研究成果也成为概率论“军火库”的重要部分。

另一方面，概率分布是否存在什么共性呢？我们的许多结论都是依赖于分布的具体类型。对于一个分布成立的结论，对于另一种分布可能并不成立。一个对任意分布都成立的结论可以大大简化我们的研究。这在自然科学和社会科学的研究中异常重要。在这些学科的研究中有许多随机变量。比如说，为了研究金矿，往往需要知道石头中含金量X的概率分布。然而，这些随机变量的分布类型不可能提前获知 (甚至于永远不能准确的知道)。这样的话，整个研究就被停在了第一步。如果我们可以得出一个对任意分布都成立的结论，那么我们就可以沿着这个结论继续进行下去。

自然有时候比我们想像的慷慨，它给出了一个概率论中相当核心的一组定律：中心极限定律(central limit theorem)。这组定律不但对于任意分布都成立，还特别提示我们：要特别注意正态分布。我们下面看看，中心极限定律是如何说的。

### 中心极限定律

先来看中心极限定律的一个版本:

随机变量X1,X2,...,Xn是相互独立的随机变量，并有相同的分布(IID, independent and identically distributed)。分布的期望为μ，方差为σ2，μ,σ都为有限值，且σ≠0。这些随机变量的均值为X¯=1n∑ni=1Xi。让ζn=X¯−μσ/n√，那么

limn→∞P(ζn≤z)=Φ(z)

其中Φ(z)是标准正态分布的分布函数。

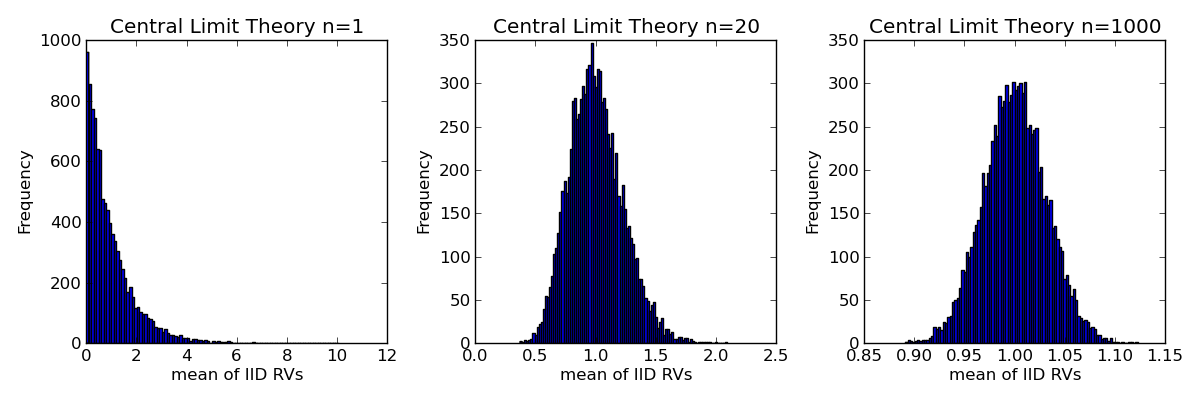
简单来说，我们寻找n个IID随机变量的均值X¯。当n趋进无穷时，这个均值(一个新的随机变量)趋近一个正态分布。

(通过ζn的变换，可以从正态分布的X¯导出标准正态分布ζn。)

### 演示中心极限定律

我们下面取n个IID随机变量，让它们都符合λ=1的指数分布，并观察它们均值的分布状况。为了观察它们的分布，我们使用随机数生成器，来进行10000次采样。即进行100000次实验，每次实验获得一组随机变量的取值，得到一个均值。总共获得10000个均值。绘制均值分布的直方图。

分三种情况，分别让n等于1，20， 100:



在第一种情况下,X¯=X1/1=X1，即X¯本身是指数分布。

在第二、三种情况下，均值的分布越来越偏离一个指数分布，分布的形状不断趋近于一个正态分布。

代码如下：

[复制代码](javascript:void(0);)

# By Vamei

# Central Limit Theory

# X is exponential distribution with lambda = 1

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import expon

# Get one sample of (X1 + X2 + ... + XN)/N

def sample\_mean(N):

# exponential distribution, with lambda = 1

one\_sample = expon.rvs(scale = 1, size = N)

return one\_sample.mean()

# Increase N: 1, 20 , 1000.

# Demo of Central Limit Theory in histogram

plt.figure(figsize=(12, 4))

for N, subp in zip([1, 20, 1000], [131, 132, 133]):

# generate samples

all\_means = np.array([sample\_mean(N) for i in range(10000)])

# plot figure

plt.subplot(subp)

plt.hist(all\_means,bins=100,color="blue")

plt.title('Central Limit Theory n=%i' % N)

plt.xlabel('sample means')

plt.ylabel('Frequency')

plt.tight\_layout()

plt.savefig('./central\_limit.png', dpi=None, facecolor='w')

[复制代码](javascript:void(0);)

练习：这段代码检验的是指数分布的均值。可以改写成检验其它分布是否符合中心极限定律，比如均匀分布的均值。

### 证明

我将使用矩生成函数来证明上面的定律。假设Xi−μ的矩生成函数为M(t)。因此，M′(t)=μ,M(2)(t)=σ2。

当n趋近无穷时，t/(σn√)趋近0。M(t)可以展开为:

M(t)=1+12σ2t2+o(t2)

o(t2)表示比t2更高阶的t的乘方。

根据矩生成函数的性质，ζn的矩生成函数写为

Mζn=[M(tσn√)]n=(1+t22n+o(t2/n))n

o(t2/n)表示，当n趋于无穷时，早于t2/n消失的项。

(根据微积分，证明从略)：当n趋近于无穷时，上面的表达式趋近：

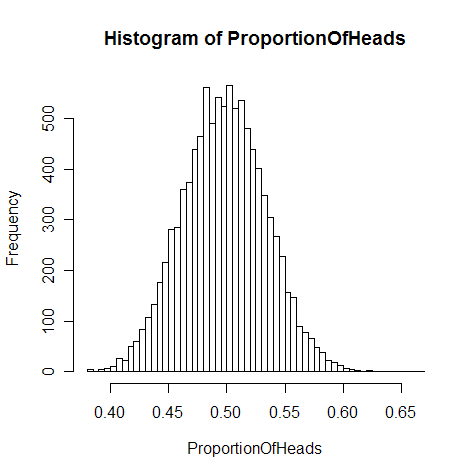
Mζn(t)→et2/2

这正是标准正态分布的矩生成函数。因此Zn的分布趋近于标准正态分布。

上面介绍的中心极限定律有一个先决条件，即产生均值的N个随机变量为IID(独立、同分布)随机变量。在其它的版本的中心极限定律中，各个随机变量可以不完全独立。事实上，中心极限定律是一个还在积极研究中的领域。

### 花边

中心极限定律的原型可以追溯到18世纪de Moivre的研究。他经过实验发现，大量正面抛硬币的话，结果(1：正面，0：反面)的均值是一个正态分布。这里，de Moivre研究的分布是多个伯努利分布的随机变量的均值。



硬币投掷：均值的分布

(想像一下，当时没有计算机，更别说随机数生成器了。为了检验结果，de Moivre真的投了几千次硬币…… 数学家是很神奇的动物)

为了更加直观的理解中心极限定律的结果。我们来设想一下，如果一个大米缸中混装了黑白两种米，各占一半。从中随便抓一把，这一把中有n个米粒。如果n比较小的话，那么很有可能出现一些极端值，比如n = 3，出现三个纯白的米粒。但是，如果“一把”很大，比如1000颗米粒，那么出现1000个米都是白色的概率很小，而白米和黑米一半一半的概率很大，也就是一个类似于正态分布的分布方式。



我们可以将中心极限定律方便的用于许多统计问题。需要注意的是，中心极限定律要求n趋近无穷。在实际应用中，我们往往让n等于一个“足够”大的数，比如上面的1000。这个数字是否足够大呢？这取决于X是什么样的分布。对于某些分布来说，均值分布趋近于正态分布的速度很慢，这要求我们采用更大的n值。

### 总结

中心极限定律（均值趋近正态分布）

# [数学与编程：“概率论”总结](http://www.cnblogs.com/vamei/p/3472301.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

终于写完[概率论](http://www.cnblogs.com/vamei/p/3178534.html)部分的内容。写一个总结，同时算是导读（似乎有些偷懒）。这两天花了些时间，对原文进行修订。自己预期的目标，是将概率论的理论和编程结合起来。这里选择的编程工具是Python和一些第三方的包。我看过一些概率的书籍。有一些在数学上处理得好，但太偏重于纯粹数学；概率和编程结合的书，又过“实用”，没有把概率论的数学本质讲清楚。理论是最好的实用。为什么不能把简洁优美的数学理论和同样美妙的编程合在一起呢？有了这个疑问，也就有了这一系列文章的探索。

回头看看，Python在统计概率方面还无法完全和R语言比拟。比如写双变量高斯分布的时候，Scipy里没有现成的函数。可能选用R语言的话，功能会更全面。未来如果有时间的话，希望可以为这一系列文章增加一个R语言的实现版本。

总的来说，概率论的体系是比较简单的。我觉得可以归纳为下面几个点：1. 公理体系，2. 典型分布，3. 描述量，4. 普适定律。在概率论的系列文章，或者是任何概率论的教材中，都会涉及这四个方面。

概率论的难点可能在于它的公理体系。它的叙述非常简单，但不容易把握。要知道，在概率论初期发展的一两百年里，公理体系并不存在。当时的概率论研究很碎片化。公理化体系是建立在现代数学的基础上，特别依赖于集合论和测度论的发展。因此它等到20世纪才建立起来。一些基本的概率论术语，如样本、实验、事件、随机变量，放在集合论的大环境里，就比较容易理解。公理化体系的难点在于概率测度的概念，也就是通俗说的“概率”。测度论本身为了探索“长度”、“面积”和“体积”这样一些代表集合“大小”而产生的数学学科。概率论是在事件集合上，增加一个称为“概率”的测度，对应起来。概率测度代表事件这样一个集合的“大小”。集合越大，越可能出现。我们随后引入的一些概念和工具，比如条件概率和连续分布，都是建立在概率测度的基础上的。所以，尽管测度论一般被认为是一门抽象学科，但一定程度的测度论知识非常有助于概率论的学习。

概率论的典型分布大多来源于古典研究，也就是出现在公理体系之前。这些典型分布大多基于一些具体的现实过程，比如连续打靶10次，中靶次数的分布。它们对应现实过程的一些经典事件的分布，因此非常实用。我们在研究许多概率论问题时，并不需要从最基本的公理从头开始，而更多的是根据问题的特性，来直接引用具体分布的结论。因此，这些经典分布就像一本“电磁学千题解”，我们需要掌握的，是问题所在的具体页码。这些经典分布最早都是为了解决概率论中的难题产生的。最经典和早期的一些数学分布来自于赌博这一现实活动。比如最早的概率论研究就来自于帕斯卡和费马的一系列通信，而这些通信都是为了解决帕斯卡的赌徒朋友所碰到的问题。数学的许多理论体系都来自于解题过程中产生的思想，比如高次多项式解这一数学难题，就孕育了抽象代数中的许多重要概念。如果没有了数学难题，比如千禧年七大问题，数学就丧失了它赖以生存的动能。

描述量有很强的实用意义。通过参照典型分布，我们已经可以大大简化工作量。但作为需要作出决定的普通人，我们往往只需要最重要的一些结果来参考。期望和方差就可以提供很重要、又很容易把握的分布信息。之前有一个报道，是美国一对老夫妇发现某种彩票的期望在一些特殊轮，买家的期望值是正值。他们于是根据期望来买彩票，结果不声不响的赚了几百万美元，还开公司教别人买彩票（事后证明，彩票公司的设计有问题）。下次打三国杀的时候，可以尝试考虑一下判定牌的期望（但期望不是具体的结果，除非打得够多 :-)，才比较容易和期望值吻合。）。再比如说，作为股票买家，很容易分辨股价比较“震荡”和股价比较“平静”的时期，这就是方差的概念。在理论上，我们可以继续求其它的描述量，它们也有各自的具体意义，比如斜度。但无疑，期望和方差是生活中最常遇到，也最常处理的一些描述量。在矩概念上建立起的矩生成函数，是一种很常用的概率论理论研究工具。

最后，是一些普适定律，比如[Chebyshev不等式](http://www.cnblogs.com/vamei/p/3232313.html)和[中心极限定律](http://www.cnblogs.com/vamei/p/3460965.html)。这些定律不依赖于具体分布的类型，对所有分布都成立。这些定律就好象拿木棍做兵器，虽然糙了些，但胜在可以就地取材。但如果真的没有进一步信息，有一根木棍总胜过赤手空拳。这些定律可以参考它们的具体叙述。

概率论是许多学科的基础，比如我们后面要进入的统计。统计是对一大票人口的数据的研究。统计的“概率性”来自于，许多时候，我们无法获得所有人口的具体信息（由于收集信息的困难），我们需要通过采样来推断整个人口的信息。这样一种管中窥豹的研究方法，就具有很强的概率性。比如我们想了解程序员的比例，可以通过对500人采样来进行估计。尽管总人口中的程序员是确定的，但这500人的采样中，有多少个程序员，是一个概率事件。

概率论同样是随机过程的基础。随机过程的学习特别有助于理解一些具体分布的生成过程。此外，随机过程在计算机科学中也很重要，比如scheduler的线程，可以看作是随机的排队过程。我们可以通过对随机过程的了解，来优化scheduler。

程序员并不需要特别高深的数学，但数学可以是程序员武器库中一样有力的武器。在我觉得，抛开需要一定直觉和运气的数学研究，数学本身完全可以看作一个固定的、有确定规则的思维“编程”工具，它和程序员平常钻研的“操作系统”、“编程语言”，有很大的类似性：基于一定的规则，并沿着这样的规则，可以有一个确定的结果。计算机的一些领域是和数学紧密相关的，比如算法、函数式编程、计算机图形、数据挖掘、机器学习。许多编程书籍中“附赠”的数学节选，往往太过碎片化。看起来提供了所需要的数学，但将整个数学体系抛到脑后，只取结论。因为这个跳跃，让本身不是特别复杂的数学显得异常高深。其实，程序员这个异常好学的群体，为自己增加一个新的技能——数学，也是为自己打开许多计算机领域的门。

希望读过这一系列文章，可以“新技能get”。OK，下一步，统计。

# [线性代数01 线性的大脑](http://www.cnblogs.com/vamei/p/3496890.html)

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢!

线性代数是一门大学课程，但也是相当“惨烈”的一门课程。在大学期间，我对这门学科就没怎么学懂。先是挣扎于各种行列式、解方程，然后又看到奇怪的正交矩阵、酉矩阵。还没来得及消化，期末考试轰然到来，成绩自然凄凄惨惨。

后来读了更多的线性代数的内容，才发现，线性代数远不是一套奇奇怪怪的规定。它的内在逻辑很明确。只可惜大学时的教材，把最重要的一些核心概念，比如线性系统，放在了最后。总结这些惨痛的经历，再加上最近的心得，我准备写一些线性代数的相关文章。

这一系列线性代数文章有三个目的：

1. 概念直观化
2. 为“[数据科学](http://www.cnblogs.com/vamei/p/3178534.html)”系列文章做准备，没有线性代数基础，没法深入统计和机器学习。
3. 线性代数运算的代码实现。这是经典的程序员挑战。参看[一天能学会的计算机技术](http://www.cnblogs.com/vamei/p/3458615.html)

线性代数是现代数学、自然科学的基础工具。在计算机领域，数据挖掘、机器学习、图形处理，数值运算这几块儿都与线性代数紧密相关。如果你对这些技术感兴趣，这些线性代数的文章可以作为你的参考读物。

这一篇，我将引入线型代数的核心：线性系统。让人惊奇的是，这一核心概念，早就根植在我们的思维中。

### 生活中的线性：超市结算

我们想象一个只卖两个商品的超市，销售青菜、黄豆。青菜每捆5元，黄豆每盒3元。此外，这个超市还有个积分系统，每捆青菜积分2分，每包黄豆积4分。需要一个结算系统，为客户计算总价和积分。



超市结算

这对程序员来说不算挑战。每个语言都可以轻松的实现，比如用Python:

# By Vamei

def bill(x1, x2):

y1 = 5\*x1 + 3\*x2

y2 = 2\*x1 + 4\*x2  
 return y1, y2

x1，x2分别为青菜和黄豆的数目。y1，y2为总价和积分。通过输入不同品种的购买数目，我们得到输出。这里的输出有两个元素：总价和积分。

上面的计算，还可以写成一组简单的数学方程：

y1=5×x1+3×x2

y2=2×x1+4×x2

我们试想这样一种情况：一对夫妻去超市买菜。丈夫买了1捆青菜，2盒黄豆，结账的时候，为11元和10个积分。妻子买了2捆绑青菜，3盒黄豆，结账的时候，为19元和16积分。

但如果妻子结账前碰到丈夫了，俩人把东西放在一起，总共3捆青菜，5盒黄豆。按照我们的结算系统，总价为5×3+3×5=30元，总积分为2×3+4×5=26积分。

你可能会反驳我，为什么要那么麻烦呢？把刚才的两个单子加在一起不就可以了。11+19=30元，10+16=26积分。这通过结算系统的计算结果完全相同。

这想法没错。你已经在运用线性系统(Linear System)的思维了：

几个购物车里的东西，分开结账的几张小票的总和，和一次算总帐的结果相同。

线性系统还有更复杂的情况。把两个购物车给销售员，让销售员按相同的配比，丈夫的来3车，妻子的来2车。那么，新的总价，应该是丈夫的小票乘3，加上妻子的小票乘2。

线性的思维方式是如此的普遍，以致于我们要多想一下，才能想出非线性的例子。下面是一个非线性的情况：超市更改积分系统，积分超过20的话，将获得双倍积分。这个时候，如果分开结账，丈夫和妻子的积分都不到20，那么积分分别为10和16，总和为26。而合在一起结账，由于积分超过了20，积分将是52。有生活经验的夫妻们，一定是合在一起结账，而不是分开结账了。

我们创造了一个非线性的系统。把这个新的结算系统编成函数，依然用Python：

[复制代码](javascript:void(0);)

# By Vamei

def non\_linear\_bill(x1, x2):

y1 = 5\*x1 + 3\*x2

y2 = 2\*x1 + 4\*x2

if y2 > 20:

y2 = y2 \* 2

return y1, y2

[复制代码](javascript:void(0);)

非线性并不是人们的惯常思维方式。超市和商场常有复杂的打折、赠券、积分系统， 这些系统很多时候是非线性的。大脑需要耗费很大能量，才能处理得过来。于是，作为超级线性的男生，我通常的想法都是：去它妈的，老子不要那么麻烦的合单或拆单了。

(奇怪的是，妹纸可以超级熟练的处理各种非线性的购物系统，甚至并行处理多个。上帝拿走的那根肋骨，一定是非线型的……)

### “一个”

我们即将要改变我们对一个单位的数据的理解。举出一个数据

做为程序员，最直接会列举出一个数据，比如一个整数，一个浮点数。

那一个结构体呢？C语言中的结构体可以包含有多个元素。我们知道，每个元素分开写出来，并不是结构体的完整数据。比如：

typedef struct {

int veg;

int bean;

} Cart;

再继续，一个对象的数据呢？一个对象可以有多个属性。当我们说一个对象的数据时，我们指的是这个对象的多个属性。比如：

public class Cart{

int veg;

int bean;

}

再比如，我们在说一个人的数据时，包括姓名，身高、体重、IQ多个值。这多个值可以构成这个人的“一个”数据。我们可以在SQL数据库中建立这样一个Person(name, height, weight, IQ)的表。每一行，也就是一个记录(record)，算是一个数据单位。

即使是列表这样的数据容器，如果固定每个位置数据的意义，那么一个列表也可以算是“一个”数据。比如丈夫购物车为[1，2]，妻子的购物车为[2，3]。

这种包含了多个元素的数据，称为向量(vector)。与之对应，一个单一的数值，称为标量(scalar)。



 一个向量

我们用带小箭头字母表示，来表示一个向量。比如丈夫的购物车：

x⃗ =[12]

向量可以相加减，这时只需要对应行的元素相加就可以，相当于合并或分开购物车。比如丈夫和妻子的购物车合并：

[12]+[23]=[35]

向量也可以与一个标量相乘。比如x⃗ ×5表示5个购物车的量。这时只需将标量与向量的各行元素相乘。

5[12]=[510]

伴随着向量，有一个简单的概念，即维度(dimension)。上面的购物车向量，包含了两个数值，即青菜的数目和黄豆的数目。我们因此说该向量是二维的。而结构体中元素的个数、对象的属性个数，都是维度。我会在以后的文章中深入维度这一概念。

有了对数据的深入理解，那么线性系统的特点可以总结如下：

L(aD1→+bD2→)=aL(D1→)+bL(D2→)

D1→和D2→是向量，分别是丈夫和妻子的购物车。而a, b为两个标量，比如a为2，b为3，表示丈夫那样的购物车乘2，妻子的购物车乘3。L为结算系统。方程右边表示，合在一起结账。方程右边表示，丈夫和妻子分开小票，相乘再相加。方程的两边相等。

### 矩阵革命

在数学上，我们已经有一组方程表示出了一个线性系统。上面的方程组有些不方便的地方：

* 输入的元素(黄豆数目)和系统参数(单价)混合在一起
* 有很多字母

数学家是偷懒的动物，这点和程序员很像。他们最后找到了一种省事的记述方式。利用刚才的向量。分离的表示输入、线性系统和输出的关系：

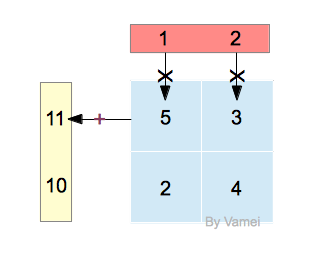
[1110]=[5234][12]

方程最左是个向量，最右是个向量。奇怪的是中间用括号括住的一堆数字。这被称为一个矩阵(Matrix)。可以看到，这个矩阵中有四个元素，包含了各个物品的单价和各个物品可获得的积分。这通常是结算系统所包含的数据。我们可以猜测到，这个矩阵相当于一个结算系统。左边的向量是输出，右边的向量是输入。



结算系统

这个结算系统运作时，把输入向量放横，再和结算系统的每一行元素分别相乘，即获得对应的输出元素。比如输出的第一个元素：



根据这一运算规则，一个线性系统就完全用一个矩阵表示出来了。

可以把矩阵表示成字母A，那么用代数的形式，写出输出和矩阵、输入的关系：

y⃗ =Ax⃗

这个代数形式，在线性代数中，有基础性的地位。方程的右边，我们说矩阵和向量进行了“乘法”运算。这一运算的规则，是按照我们上面所描述的那样运行的。这简直是对乘法符号的一次“运算符重载”(operator overload)。

我们可以用程序来实现上面的计算过程。编写类似的C程序并不复杂。更方便的是调用现有的库函数，比如Python中的numpy:

[复制代码](javascript:void(0);)

# By Vamei

import numpy as np

# matrix

a = np.matrix([[5, 3],[2, 4]])

# input Vector

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

# multiplication

y = np.dot(a, x)

print(y)

[复制代码](javascript:void(0);)

矩阵这个东西把结算系统的表示方式大大缩减。更重要在于，线性系统和矩阵是互通的。矩阵表示的是一个线性系统。一个线性系统也总可以表示一个矩阵(证明从略)。

绕了半天，矩阵 = 线性系统。

### 总结

线性代数的核心是线性系统的概念。线性系统与矩阵的等同性，让线性代数后面的内容，转入到对矩阵的研究中。但核心要牢记。

线性系统的概念在生活中非常常见。人的思维很多时候也是线性的。思考生活中线性和非线性的例子。

广义的数据可以表示成多维的向量。