

用機率分布描述亂中有序的世界 - [SEP]離散型分配 (2)



簡報閱讀



範例與作業



問題討論



學習心得(完成)

重要知識點

回憶一下昨天交的機率定義

負二項分配

例子：如果想要了解要擲多少次骰子，才會出現...



重要知識點






今天的課程，介紹離散型分配的其他分配，包

- 超幾何分配(Hypergeometric Distribution)

今天教的例子中，可以用來計算大樂透的機率。

回憶一下昨天交的機率定義

定義	
隨機實驗(Random Experiment)：每次實驗結果都具有不確定性，透過實驗，能發現某種規律性	
樣本空間 (Sample Space; S)：一項試驗中所有可能發生結果所形成的集合。	
樣本點(Sample)：樣本空間的每一元素，稱為一個樣本點。	 or 
事件(Event; E)：樣本空間中每一部分集合(包括空集合)均為此一樣本空間之一事件(即每一可能發生的結果)	事件1：骰出數值小於2 事件2：骰出數值為奇數
試驗 (Trial)：一個隨機實驗的執行一次	丟一次骰子，就是一次實驗。

負二項分配

負二項分佈是統計學上一種描述在一系列獨立同分佈的伯努利試驗中，X為成功次數到達指定次數(記為k)時，需要試驗的機率分布，表示為 $X \sim NB(k, p)$

$X \sim NB(k, p)$

k：是成功的次數

p：是成功的機率

$$P(X = x) = C_{k-1}^{x-1} p^k (1 - p)^{x-k}$$


$x = k, k + 1, k + 2, \dots$

累積機率函數	nbinom.cdf
樣本點	nbinom.rvs
統計量計算	nbinom.stats

例子：如果想要了解要擲多少次骰子，才會出現紅色？


X：第1次出現紅色時，共擲的次數

實驗1



$X = 1$

實驗1



$X = 3$

X：第k次出現紅色時，共擲的次數

若進行x次才停止，由於最後一次為成功，且第一次至第x-1次中，有k-1次成功，x-k次失敗，排列組合之後，X的機率函數為

$$X \sim NB(k, p)$$

$$P(X = x) = C_{x-1}^{k-1} p^k (1-p)^{x-k}, x = k, k+1, k+2, \dots$$

問題轉換：

丟n次出現紅色的次數 --> 第k次出現紅色時，共擲的次數

binomial (n,p) 轉換成負二項分配NB(k,p)

超幾何分配

記做： $X \sim \text{HG}(N, K, n)$

N ：共有幾個物件， $N = 0, 1, \dots$

K ： N 個物件中，有 K 個是你關心的物件類型個數， $K=0, 1, 2, \dots, N$

n ： K 個物件，要抽出 n 個物件， $n=0, 1, \dots, N$

$$P(X = x) = \frac{\binom{K}{x} \binom{N-K}{n-x}}{\binom{N}{n}},$$

$$x = \max\{n - N + K, 0\}, \dots, \min\{K, n\}$$

$$E(X) = np, p = \frac{K}{N}$$

$$\text{Var}(X) = np(1 - p) \frac{N - n}{N - 1}$$

超幾何分配	Python 語法(scipy.stats)
機率質量函數	hypergeom.pmf
累積機率函數	hypergeom.cdf
樣本點	hypergeom.rvs
統計量計算	hypergeom.stats

例子：現在有兩堆骰子，30個為紅色數字，20個為黑色數字，取出10個，有3個是全紅有其分配規律性？

超幾何分配描述了由有限個物件中抽出 n 個物件，成功抽出指定種類的物件的個數 (不歸還(without replacement))，若 $n=1$ ，超幾何分布還原為伯努利分布 (bernulli(p))。

若隨機變量 X 服從參數，則記為 $H(n, K, N)$ ，

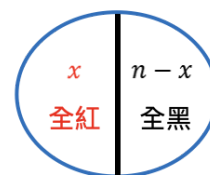
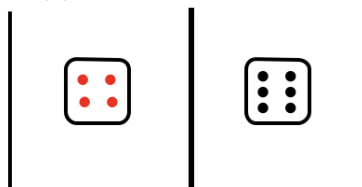
N ：共有幾個物件， $N = 0, 1, \dots$

K ： N 個物件中，有 K 個是你關心的物件類型個數， $K=0, 1, 2, \dots, N$

n ： K 個物件，要抽出 n 個物件， $n=0, 1, \dots, N$

30(K) 個全紅骰子

20 個全黑骰子



超幾何分布(cont.)

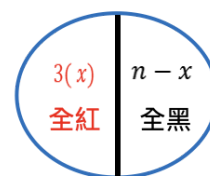
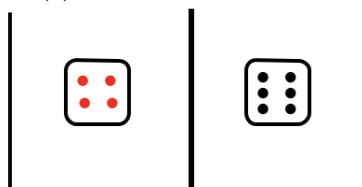
現在有兩堆骰子，30個為紅色數字，20個為黑色數字，取出10個，有3個是全紅的要怎麼透過超幾何分配計算？

有50(N)個骰子

抽出10(n)個

30(K) 個全紅骰子

20 個全黑骰子



$$P(X = 3) = \frac{\binom{30}{3} \binom{50-30}{10-3}}{\binom{50}{10}}$$

常見離散型分配表格

分配	符號	X所代表意思
離散均勻分配 uniform		
伯努利分配 Bernuli	$X \sim \text{bernoulli}(p)$	丟一次骰子出現的紅色的次數
二項分配 Binomail	$X \sim \text{binomail}(p)$	
負二項分配 Negative binomail	$X \sim NB(k, p)$	第 k 次出現紅色時，共擲的次數
超幾何分配 hypergeometric distribution	$H(n, K, N)$	由有限個物件中抽出n個物件，成功抽出指定種類的物件的個數

知識點回顧

- 生活中常見的離散分配的意義與性質，包含五種離散分配，包含離散均勻分配、伯努利分配、二項分配、負二項分配與超幾何分配。
- 運用python語法了解各分配的特性與應用 pmf、cdf、rvs和 stats

延伸閱讀

scipy.stats 離散分配官方教學

負二項分配

網站：[scipy.stats.nbinom](https://docs.scipy.org/doc/scipy/reference/stats.html#negative-binomial)

scipy.stats.nbinom

`scipy.stats.nbinom(*args, **kwargs) = <scipy.stats._discrete_distns.nbinom_gen object>` [\[source\]](#)

A negative binomial discrete random variable.

As an instance of the `rv_discrete` class, `nbinom` object inherits from it a collection of generic methods (see below for the full list), and completes them with details specific for this particular distribution.

Notes

Negative binomial distribution describes a sequence of i.i.d. Bernoulli trials, repeated until a predefined, non-random number of successes occurs.

The probability mass function of the number of failures for `nbinom` is:

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

for $k \geq 0$.

`nbinom` takes n and p as shape parameters where n is the number of successes, whereas p is the probability of a single success.

The probability mass function above is defined in the "standardized" form. To shift distribution use the `loc` parameter. Specifically, `nbinom.pmf(k, n, p, loc)` is identically equivalent to `nbinom.pmf(k - loc, n, p)`.

Examples

```
>>> from scipy.stats import nbinom
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots(1, 1)
```

Calculate a few first moments:

```
>>> n, p = 0.4, 0.4
>>> mean, var, skew, kurt = nbinom.stats(n, p, moments='mvsk')
```

Display the probability mass function (pmf):

```
>>> x = np.arange(nbinom.ppf(0.01, n, p),
...               nbinom.ppf(0.99, n, p))
>>> ax.plot(x, nbinom.pmf(x, n, p), 'bo', ms=8, label='nbinom pmf')
>>> ax.vlines(x, 0, nbinom.pmf(x, n, p), colors='b', lw=5, alpha=0.5)
```

Alternatively, the distribution object can be called (as a function) to fix the shape and location. This returns a "frozen" RV object holding the given parameters fixed.

Freeze the distribution and display the frozen pmf:

```
>>> rv = nbinom(n, p)
>>> ax.vlines(x, 0, rv.pmf(x), colors='k', linestyle='-', lw=1,
...          label='frozen pmf')
>>> ax.legend(loc='best', frameon=False)
>>> plt.show()
```

scipy.stats.hypergeom

`scipy.stats.hypergeom(*args, **kwargs) = <scipy.stats._discrete_distns.hypergeom_gen object>` [\[source\]](#)

A hypergeometric discrete random variable.

The hypergeometric distribution models drawing objects from a bin. M is the total number of objects, n is total number of Type I objects. The random variate represents the number of Type I objects in N drawn without replacement from the total population.

As an instance of the `rv_discrete` class, `hypergeom` object inherits from it a collection of generic methods (see below for the full list), and completes them with details specific for this particular distribution.

Notes

The symbols used to denote the shape parameters (M , n , and N) are not universally accepted. See the Examples for a clarification of the definitions used here.

The probability mass function is defined as,

$$p(k, M, n, N) = \frac{\binom{n}{k} \binom{M-n}{N-k}}{\binom{M}{N}}$$

for $k \in [\max(0, N - M + n), \min(n, N)]$, where the binomial coefficients are defined as,

$$\binom{n}{k} \equiv \frac{n!}{k!(n-k)!}.$$

The probability mass function above is defined in the “standardized” form. To shift distribution use the `loc` parameter. Specifically, `hypergeom.pmf(k, M, n, N, loc)` is identically equivalent to `hypergeom.pmf(k - loc, M, n, N)`.

Examples

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

Suppose we have a collection of 20 animals, of which 7 are dogs. Then if we want to know the probability of finding a given number of dogs if we choose at random 12 of the 20 animals, we can initialize a frozen distribution and plot the probability mass function:

```
>>> [M, n, N] = [20, 7, 12]
>>> rv = hypergeom(M, n, N)
>>> x = np.arange(0, n+1)
>>> pmf_dogs = rv.pmf(x)

>>> fig = plt.figure()
>>> ax = fig.add_subplot(111)
>>> ax.plot(x, pmf_dogs, 'bo')
>>> ax.vlines(x, 0, pmf_dogs, lw=2)
>>> ax.set_xlabel('# of dogs in our group of chosen animals')
>>> ax.set_ylabel('hypergeom PMF')
>>> plt.show()
```

[下一步：閱讀範例與完成作業](#)

