Approximating Poisson Distribution: An Monte Carlo Approach

Abstract

In this notebook, we illustrate how to construct an approximate distribution of Poisson Distribution, we mainly use the Monte Carlo method, we will demonstrate that after many many times of experiments, using the sample data collected from the experiments, we can derive a pretty good approximation of Poisson Distribution which has only a subtle differences in C.D.F. and P.D.F. when the amount of trials is large enough.

Poisson Distribution

A Poisson Distribution describes the distribution of the amount of happens of a specific event within a period, e.g., how many cars just passed a road in 1 hour, how many customers just enter a bank in 30 minutes, how many particles will burst out a radiative source in a period, let's say Xcars will pass that road in 1 hour, then, the random variable X obeys a Poisson Distribution, denoted as

$X \approx \text{PoissonDistribution}[\lambda]$

where λ is the hourly average traffic of that road, then given an non-negative integer x, we can compute the probability of "4 cars pass that road in 1 hour", which is

$\Pr(X = 4) = \frac{\lambda^k e^{-\lambda}}{k!} = \left(10^4 e^{-10}\right) / 4!$

let's say, in average, there has 10 cars pass that road, that is $\lambda = 10$, so that the probability of "4 cars pass that road in 1 hour" is $\left(10^4 e^{-10}\right) / 4!$. Now hopefully you see the basics of Poisson Distribution.

How to Approximate a Poisson Distribution

Let's say that We don't know the exact expression of the cumulative distribution function of Poisson Distribution, and We only know that there's a random variable X which obeys a Poison Distribution of some parameter λ, i.e. $X \approx \text{PoissonDistribution}[\lambda]$, given only this, how can we compute the probability $\Pr(X \leq x)$, where x is a non negative integer ?

Shall notice that there is a basic assumption when We say some random variable X obeys a Poisson Distribution, it is usually assumed that in every interval, the mean amount of happens of that specific event does not change, simply speaking, let's take the car-pass-road case for example, we assumed that no matter which hour in a day is, the expectation of the amount of times the car pass that road in that hour are unique, say that λ, the parameter of Poisson Distribution, is $\lambda = 4$, which equivalently means, that, the average traffic in 0 am is 4, and the average traffic in 1 am is 4, and the average traffic in 2,3,4,5, until 22, 23 hour of that day is 4, no matter which hour it is. This is what that basic assumption means. Hopefully you understand.

So, it that basic assumption useful? Yes it is useful. In Monte Carlo way, let’s say that there are *n* intervals in total, that is, whole time range can be split into to *n* parts, and in each interval, the event are “expected” to happen λ times, which is the parameter of a Poisson Distribution, also the “mean traffic” in the car-pass-road case as we denoted, so after all *n* intervals gone, the event are “expected to” happened nλ times altogether. Then, in Monte Carlo’s point of view, each happening of the event (there are nλ happenings altogether) can randomly “choose” which interval it “wanted to” happen at, let’s take an example, say there is a array of event, which’s length is nλ

events[] = [event[1], event[2], ..., event[n λ]]

Then, for each event in the array events[], say, firstly, it is event[1], then event[1] will randomly choose a interval, in

intervals[] = [interval[1], interval[2], ..., interval[n]]

say that event[1] just randomly picked interval[4], then event[2] randomly picked interval[7], then event[3] randomly picked interval[2], and so on until all event in events[] picked their own interval[i] to happen, then, for each interval in intervals[], we can counts that how many event[i]s just fallen into it, for example, event[2], event[4], event[7], event[11] all take interval[6], then, it will means that there are 4 event[i]s fallen into interval[6], we can count that, it is easy. Same operation perform on every interval, we can have such an table which maps interval[i] to the amount of event[i]s fallen to it, it may seems like below

counts = <| interval[1] -> 7, interval[2] -> 2, interval[3] -> 11, ... |>

then We only take the values from the table, we got

data = [7, 2, 11, ...]

then you will understand that this array, this array named data, shall distributed like Poisson Distribution.

Compute

First we cut the whole time range into $n = 8\,000\,000$ parts, the larger *n* is, the more accurate the result will be.

```
In[246]:= n = 100 000;
```

Just assign a concrete value to symbol λ, which is $\lambda = 4$, so that we can compute the numerical value, and will be easier to compare and plot. So, you can say that, We are gonna to approximate PoissonDistribution[λ=4].

```
In[247]:= lambda = 4;
```

Then We generate nλ random integers that evenly distributed across 1 to n, which correspond to interval[1] to interval[n] like we just said in previous section.

```
In[248]:= sampleData = Floor[RandomVariate[UniformDistribution[{0, n + 1}], n * lambda]];
                       └向…     └伪随机变数          └均匀分布
```

Counts “how many events caught” for each interval[i].

```
In[249]:= sampleDataCounts = Counts[sampleData];
                             └关联计数
```

Extract the values part.

```
In[250]:= stats = Values[sampleDataCounts];
                  └获取值
```

Check the minimum value of the experiments data:

```
In[251]:= Min[stats]
          └最小值
Out[251]= 1
```

Check the maximum value of the experiments data:

```
In[252]:= Max[stats]
          └最大值
Out[252]= 16
```

Construct the empirical distribution using data we just collected, the array named stats.

```
In[253]:= empiricalDistribution = EmpiricalDistribution[stats];
                                  └经验分布函数
```

Try some value, to see the Pr($X \leq x$) computed from the empirical distribution.

```
In[254]:= N[PDF[empiricalDistribution, {0, 1, 2, 3, 4, 5, 6, 7}]]
          └·· └概率密度函数
Out[254]= {0., 0.0765773, 0.146121, 0.198467, 0.199588, 0.158975, 0.107995, 0.0602567}
```

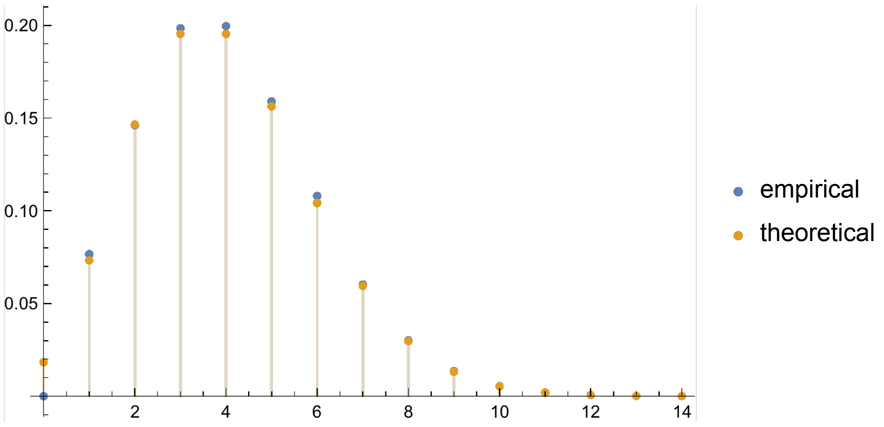
Try some value, to see the Pr($X \leq x$) computed PoissonDistribution[λ], which is, of course, theoretical.

```
In[255]:= N[PDF[PoissonDistribution[lambda], {0, 1, 2, 3, 4, 5, 6, 7}]]
          └·· └…    └泊松分布
Out[255]= {0.0183156, 0.0732626, 0.146525,
           0.195367, 0.195367, 0.156293, 0.104196, 0.0595404}
```

Compare the plot of the empirical P.D.F. and the theoretical P.D.F..

```
In[256]:= DiscretePlot[
          离散图
           {PDF[empiricalDistribution, x], PDF[PoissonDistribution[lambda], x]},
            概率密度函数                      ⋯   泊松分布
           {x, 0, 14}, PlotLegends → {"empirical", "theoretical"}]
                       绘图的图例
```

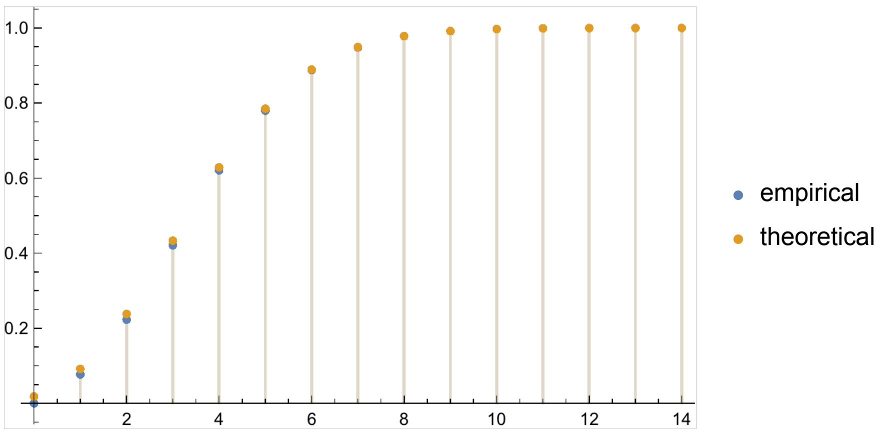
Out[256]=



Compare the plot of the empirical C.D.F. and the theoretical C.D.F..

```
In[257]:= DiscretePlot[
          离散图
           {CDF[empiricalDistribution, x], CDF[PoissonDistribution[lambda], x]},
            累积分布函数                      ⋯   泊松分布
           {x, 0, 14}, PlotLegends → {"empirical", "theoretical"}]
                       绘图的图例
```

Out[257]=



We can find that, there are only subtle differences when n are so large comparing the C.D.F. or P.D.F. between the empirical distribution which we derive it from the experiments data and the theoretical PoissonDistribution[λ], two distributions are nearly fitted into one.

Then We may do a more precise test to see how significant those two distributions are fitted to each

```
In[263]:= dataPratical = RandomVariate[empiricalDistribution, n / 100];
                         伪随机变数
```

```
In[264]:= dataTheoretical = RandomVariate[PoissonDistribution[lambda], n / 100];
                            伪随机变数      泊松分布
```

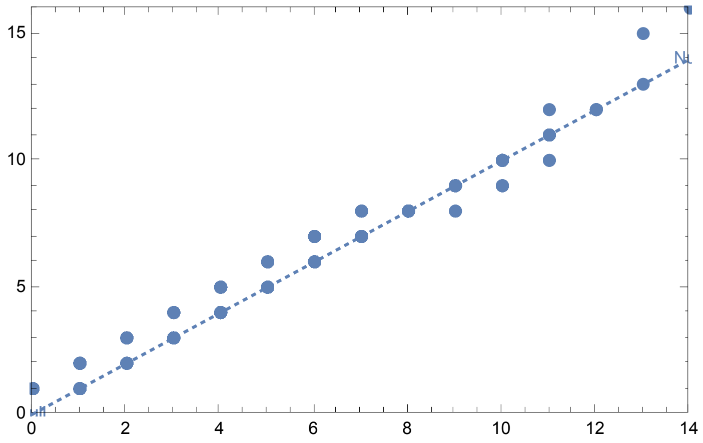
Shall notice that KolmogorovSmirnovTest might reject when it is given a very large dataset for example $n = 10\,000$, choose a smaller dataset it would be more easy to pass the test (i.e. yields a higher *p*-value so that do not reject the null hypothesis H_0 : data as input as drawn from the given theoretical distribution.

```
In[266]:= KolmogorovSmirnovTest[dataPratical, dataTheoretical]
          柯尔莫诺夫–斯米尔诺夫检验
Out[266]= 0.887939
```

If a dataset are drawn from that theoretical distribution, draw a Q-Q plot should show that points are nearly fitted around a straight line

```
In[267]:= QuantilePlot[RandomVariate[empiricalDistribution, 10 000],
          分位数图线      伪随机变数
           PoissonDistribution[lambda], PlotMarkers → {Automatic, Small}]
           泊松分布                        绘制点的标记   自动        小
```

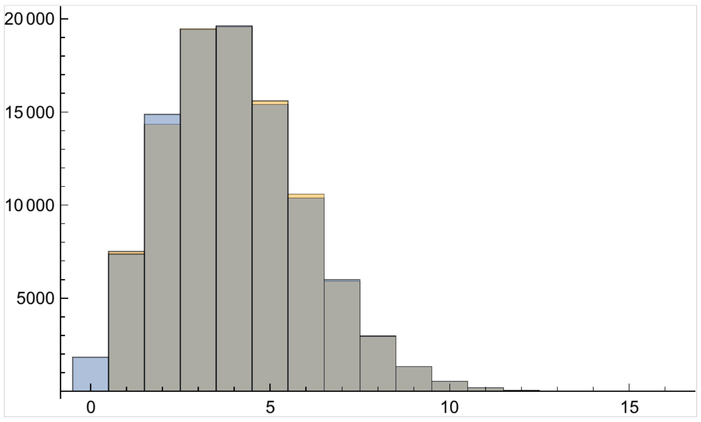
Out[267]=



So as you seen, they are fitted indeed.

```
In[268]:= Histogram[{stats, RandomVariate[PoissonDistribution[lambda], n]}]
          直方图             伪随机变数       泊松分布
```

Out[268]=



Also a histogram shows two distribution a somewhat close.

Besides, a more accurate way is to directly generate n random integers from a theoretical BinomialDistribution[nλ,1/n], you shall understand the the value each interval[i] gets obeys the same BinomialDistribution[nλ,1/n], there has n such interval[i]s, so we could directly generate n such random binomial distributed random integers

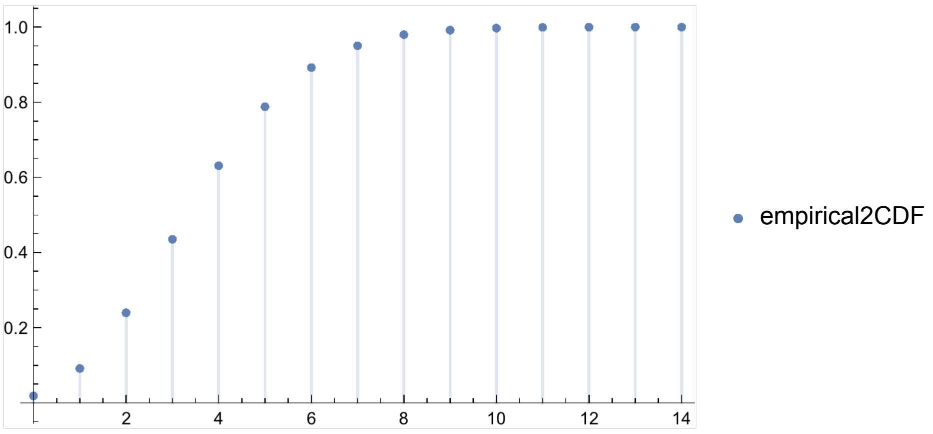
```
In[269]:= stats2 = RandomVariate[BinomialDistribution[n * lambda, 1 / n], n];
                   伪随机变数      二项分布
```

Now calculate empirical distribution from this directly generated “interval[i]s”

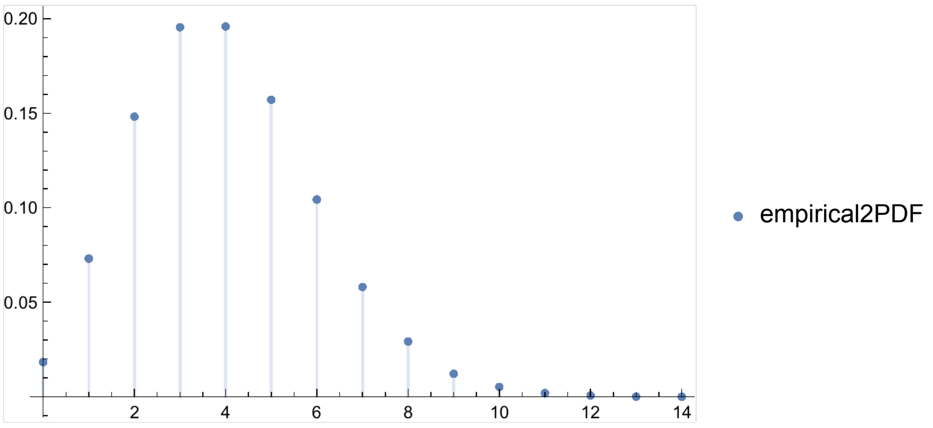
In[270]:=
```
empiricalDistribution2 = EmpiricalDistribution[stats2];
                         └经验分布函数
```

You could see that the plot is very close to a theoretical PoissonDistribution

In[271]:=
```
DiscretePlot[CDF[empiricalDistribution2, x],
└离散图        └累积分布函数
 {x, 0, 14}, PlotLegends → {"empirical2CDF"}]
             └绘图的图例
```

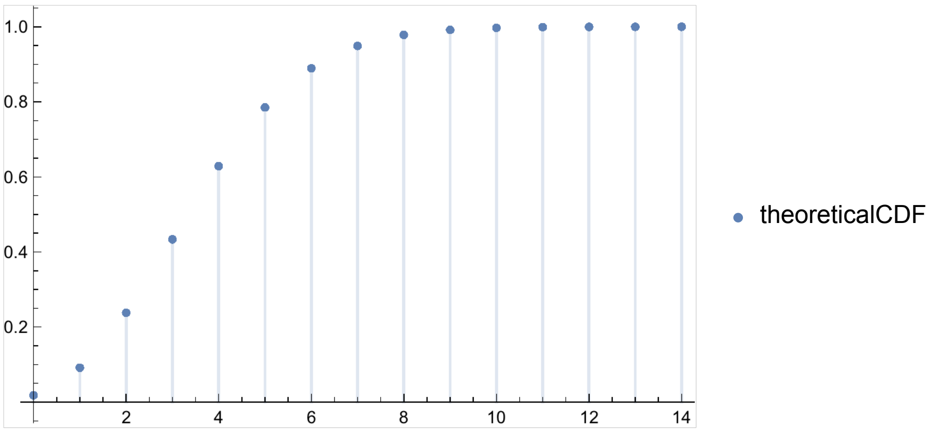
Out[271]=


In[272]:=
```
DiscretePlot[PDF[empiricalDistribution2, x],
└离散图        └概率密度函数
 {x, 0, 14}, PlotLegends → {"empirical2PDF"}]
             └绘图的图例
```

Out[272]=


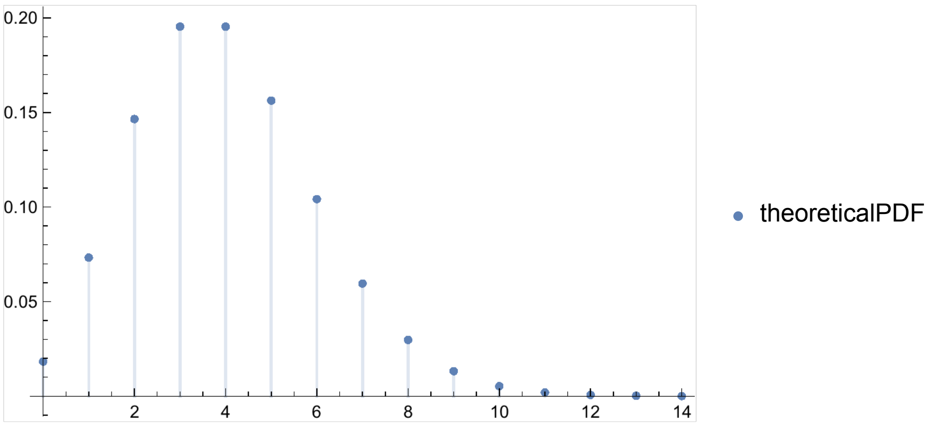
In[273]:=
```
DiscretePlot[CDF[PoissonDistribution[lambda], x],
离散图           …    泊松分布
  {x, 0, 14}, PlotLegends → {"theoreticalCDF"}]
              绘图的图例
```

Out[273]=



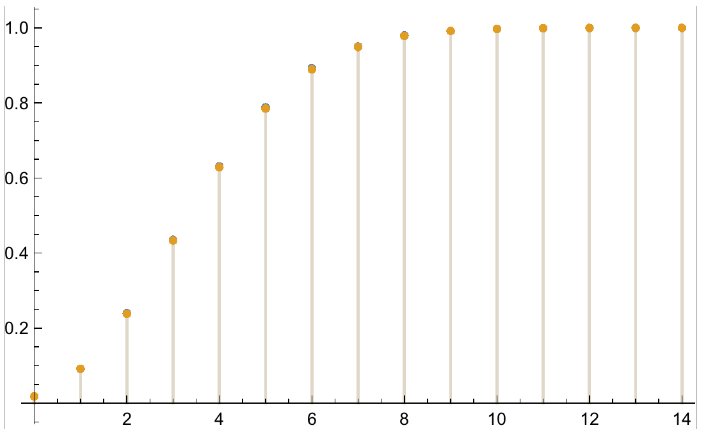
In[274]:=
```
DiscretePlot[PDF[PoissonDistribution[lambda], x],
离散图           …    泊松分布
  {x, 0, 14}, PlotLegends → {"theoreticalPDF"}]
              绘图的图例
```

Out[274]=



```
In[275]:= DiscretePlot[
          离散图
            {CDF[empiricalDistribution2, x], CDF[PoissonDistribution[lambda], x]},
             累积分布函数                      ⋯    泊松分布
            {x, 0, 14}, PlotLegends → {"empirical2", "theoretical"}]
                         绘图的图例
```

Out[275]=



Seems this empirical distribution that directly calculated from n random binomial distributed integers are even better fitting to a theoretical PoissonDistribution than the previous one.

A more accurate way to test “if data are significantly not fitted into a given theoretical distribution”---the KolmogorovSmirnovTest[data,dist], which’s null hypothesis is H_0 : the data was drawn from a population with distribution dist, and the alternative hypothesis $H_a$that it was not.

```
In[278]:= KolmogorovSmirnovTest[
          柯尔莫诺夫-斯米尔诺夫检验
            RandomVariate[empiricalDistribution2, 1000], PoissonDistribution[lambda]]
            伪随机变数                                    泊松分布
```

```
KolmogorovSmirnovTest: The p-value returned by the KolmogorovSmirnov test may not reflect the true size of
  the test for discrete test distributions.

KolmogorovSmirnovTest: Ties exist in the data and will be ignored for the KolmogorovSmirnov test, which
  assumes unique values.
```

```
Out[278]= 0.995624
```

So we should not reject the null hypothesis H_0 :input data are NOT draw from the given theoretical distribution, we could still believe the null hypothesis i.e. input data ARE draw from the given theoretical distribution therefore integers that are distributed as BinomialDistribution[nλ,1/n] are somewhat fitted into a theoretical distribution PoissonDistribution[λ]

```
In[279]:= KolmogorovSmirnovTest[
          柯尔莫诺夫-斯米尔诺夫检验
            RandomVariate[BinomialDistribution[n * lambda, 1 / n], n / 1000],
            伪随机变数       二项分布
            PoissonDistribution[lambda]]
            泊松分布

  KolmogorovSmirnovTest: The p-value returned by the KolmogorovSmirnov test may not reflect the true size of
     the test for discrete test distributions.

  KolmogorovSmirnovTest: Ties exist in the data and will be ignored for the KolmogorovSmirnov test, which
     assumes unique values.

Out[279]= 0.907804
```