

A Real Case of Applying Poisson Distribution into Software Design

by John Wang, john.innovation.au@gmail.com

with credit to Ken Liu, COO of Shenzhen Uxin Network Technology Co., Ltd.

The problem

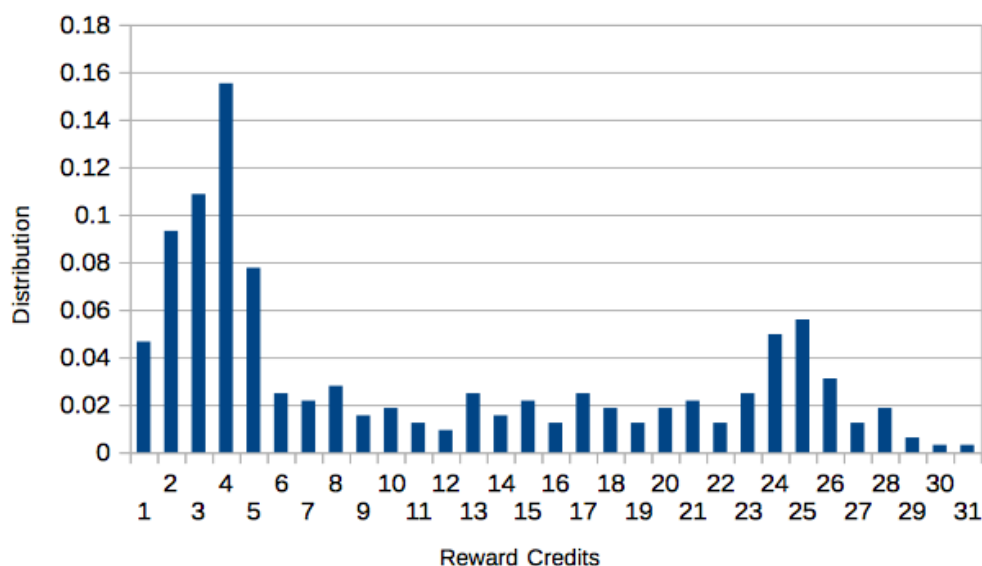
Three years ago, I received a task to work out one problem in relation to company operating cost when I worked for an Internet entrepreneurship in Shenzhen. The product of company is the most popular VoIP system in China with over 100M register users and 500K active online users everyday. There is a reward mechanism in the system that user can get random reward credits ranged from 1 to 30, if the user opened the app everyday and click “check-in” button in the app. With the credits, user can make phone call to traditional landline or other mobile phone. Yet, the landline call is part of the cost for the company, and one credit in the app can be considered as one minutes landline call. This reward mechanism is one of the company strategies to stimulate customers actively in daily using.

The COO complained that the reward algorithm in the app was terrible since it cost the company a lot of money everyday but the customers did not satisfy with the app. One of the evidence was, he got only 2 or 3 credits reward in sequence of weeks, but the overall cost of this reward feature was still as high as before it was deployed online. There might be some customers luckily enough to get very high reward, but for most other customers, they might feel depression with only 2 or 3 minutes reward.

Since the check-in feature is a very heavy duty service(nearly 500K called every day), it is not realistic to check the database (or other in-memory cache such as redis, memcache) every time individual user clicks on this button. There had been several algorithms developed to generate random credits before. Yet, all the reward credits will be recorded in database so that I suggested to find out the subtotals for each reward credits for all users. Then the data told us that the distribution of reward credits was far beyond expected.

Before optimization

average reward credits = 11.024845



Another problem is, the staff mobility is very high in Internet industry and the programming skill level of programmers might be not high enough to understand every details in the system. Thus, the requirement for the algorithm will be as beneath:

- most of users be happy with the result
- very clear, and easy to understand
- high performance
- easy to programming
- expected cost can be set up by operator with few engineering background

Solution

The application of Poisson distribution was introduced by me to solve this problem since I just completed the Probability Theory course in Shenzhen Universtiy. The Poisson distribution is an ideal mathematics tool for this problem inspite of the complexity in theory because the problem is a typical distrete probability problem. As we can see, the reward credits must be integer since the smallest unit of reward credit is minute. The value of reward credits can be 1, 2, 3, ..., but it cannot be 1.2, 0.001 etc. Thus, it is not a continuous distributions question.

Let's review the definition of Poisson distribution in statistics.

(copy from wikipedia)

“Probability of events for a Poisson distribution

An event can occur 0, 1, 2, ... times in an interval. The average number of events in an interval is designated λ (lambda). Lambda is the event rate, also called the rate parameter. The probability of observing k events in an interval is given by the equation

$$P(k \text{ events in interval}) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where

λ is the average number of events per interval

e is the number 2.71828... (Euler's number) the base of the natural logarithms

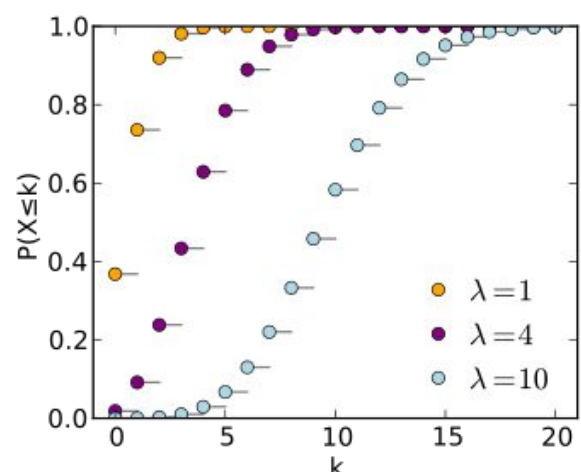
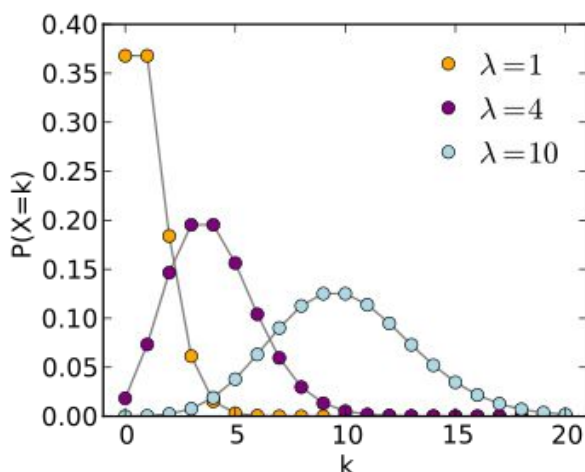
k takes values 0, 1, 2, ...

$k! = k \times (k - 1) \times (k - 2) \times \dots \times 2 \times 1$ is the factorial of k .

This equation is the probability mass function (PMF) for a Poisson distribution. ”

(end of copy)

What a horrible mathematics formular! Luckily, there are two figures from wikipedia that can help us to understand the Poisson distribution. (pictures from wikipedia)



Both figures are easier to understand. The value of λ is the expectation value of the Poission distribution in both figures. The left figure illustrates the probability of Possision distribution. On the other word, the value of y-axis is the percent of reward credits. The right figure illustrates the cumulative distribution value. That is the sum of percent of reward credits.

For example, if COO want to set the expectation of reward credits as 4 minutes, λ can be set to 4. As we can see, the purple dots in both figures are what we need.

We can pick up one random number between 0 and 1, then look up this value in the right figure to decide how many credits the user should earn. In the big picture, there are over 500K users can get reward credits based on this algorithm every day, then we have very strong confidence for average cost per user is the expected value, which is 4 in this case.

Implementation:

This idea is great! Yet, how can it be implemented? At the same time, we need to consider the skill level of programmer and operator.

The more complex the design is, the higher probability the bugs will exist. Thus, keep the design as simple as possible is a good idea. What we need is a RewardCredit class, with interfaces of setting reward range, the relationship between probability density and reward value, and get reward.

```
class RewardCredit
{
public:
    // set the minute reward range, in this case, it is 30
    setRewardRange( range );
    // set the probability density for each reward credit in range,
    // for example, 0.0000 to 0.0015 for 1 credit
    //           0.0015 to 0.0113 for 2 credits
    //           0.0113 to 0.0430 for 3 credits, and so on...
    setRewardProbability( probability density[range] );
public:
    // return the value random based on the probability set by operator
    getReward( random number );
}
```

In this way, the test case will be easy to design, and easy to valid the output of reward.

Yet, for operator, how can they decide probability density for reward? It can be part of the program to generate the Poission distribution value for particular expected value. But the problem is, it will require 3rd-party component to finish this task, and it will increase the complexity of the system. Another requirement from operating department is that they might slightly alternative some of the result. Thus, we made a decision that the operator can set the probability density manually since it is the responsibility of operator to maintain the operating cost rather than programmer.




We need a utility to generate the array of value for Poission probability density.

Is there any ready-made, reliable software to accomplish this assignment?

Yes! It can be Excel in Microsoft Office, or SpreadSheet in LibreOffice/OpenOffice.

How can we get the job done?

Suppose we need a mean value of 6.5, (yes, the expected value can be a decimal) we can input a array of number form 0 to 30. Then input the Poission formular into the formular bar for relevant cell and drag the cell to copy for all 0 to 30. We can work out all the probability mass values and cumulative distribution values by simple drag, and we can work out the expected cost for each integer from 0 to 30 for verification.

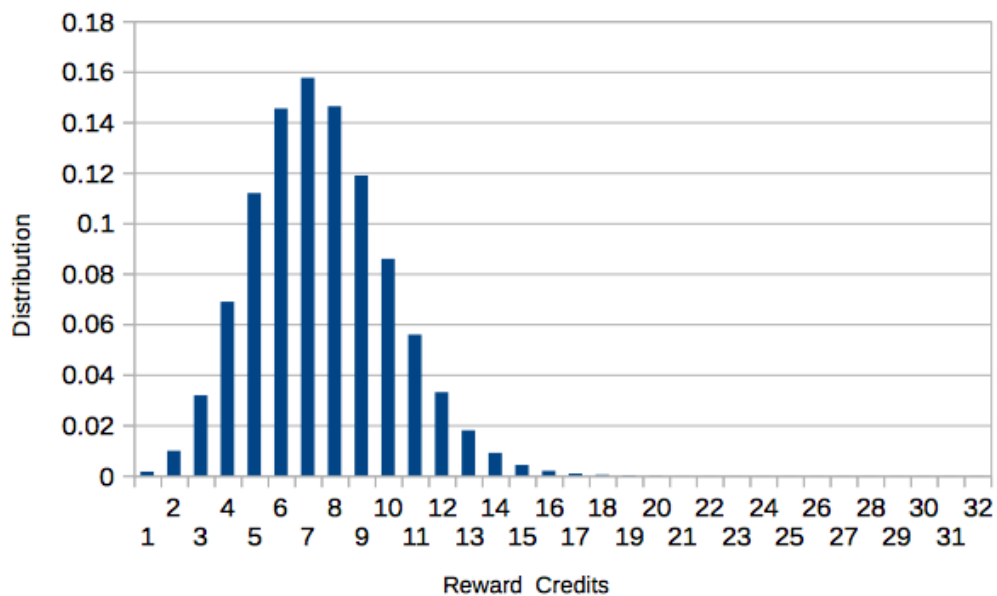
POISSON.DIST			  	=POISSON.DIST(B4,6.5,1)				
	A	B	C	D	E	F	G	H
1								
2				POISSON.DIST([B],6.5,1)		POISSON.DIST([B],6.5,0)		[F]*[B+1]
3								
4		0		0.0015034392		0.0015034392		0.0015034392
5		1		0.0112757939		0.0097723548		0.0195447095
6		2		0.0430359469		0.031760153		0.0952804589
7		3		0.1118496116		0.0688136647		0.2752546589
8		4		0.2236718168		0.1118222052		0.5591110259
9		5		0.3690406836		0.1453688667		0.8722132004
10		6		0.5265236225		0.157482939		1.1023805728
11		7		0.6727577801		0.1462341576		1.1698732609
12		8		0.7915730332		0.1188152531		1.0693372775
13		9		0.8773840493		0.0858110161		0.858110161
14		10		0.9331612098		0.0557771605		0.6135487651
15		11		0.9661204409		0.0329592312		0.3955107742
16		12		0.9839733578		0.0178529169		0.2320879196
17		13		0.9928998163		0.0089264584		0.1249704182
18		14		0.9970442434		0.0041444271		0.062166407
19		15		0.9988401618		0.0017959184		0.0287346948
20		16		0.9995697537		0.0007295919		0.0124030616
21		17		0.9998487153		0.0002789616		0.0050213087
22		18		0.9999494514		0.0001007361		0.0019139865
23		19		0.9999839138		3.44623606182549E-005		0.0006892472
24		20		0.999995114		1.12002672009329E-005		0.0002352056
25		21		0.9999985808		3.46674937171731E-006		7.6268486E-005
26		22		0.9999996051		1.02426685982557E-006		2.3558138E-005
27		23		0.9999998945		2.89466721255052E-007		6.9472013E-006
28		24		0.9999999729		7.83972370065767E-008		1.9599309E-006
29		25		0.9999999933		2.03832816217099E-008		0.00000053
30		26		0.9999999984		5.09582040542748E-009		1.3758715E-007
31		27		0.9999999996		1.22677157908439E-009		3.4349604E-008
32		28		0.9999999999		2.84786259430306E-010		8.2588015E-009
33		29		1		6.38314029757582E-011		1.9149421E-009
34		30		1		1.38301373114143E-011		4.2873426E-010
35		31		1		2.89986750078041E-012		0
36								
37						SUM(H4 to H35)		7.4999999999

The sum for verification is different? Yeah, the initial element of Poission distribution is ZERO, but in our application, the reward credit must be from 1 to 30 so that we need to do a little adjustment. For example, we can set the value to 6.5 in Poission formular if we need the expected value to be 7.5, and so on.

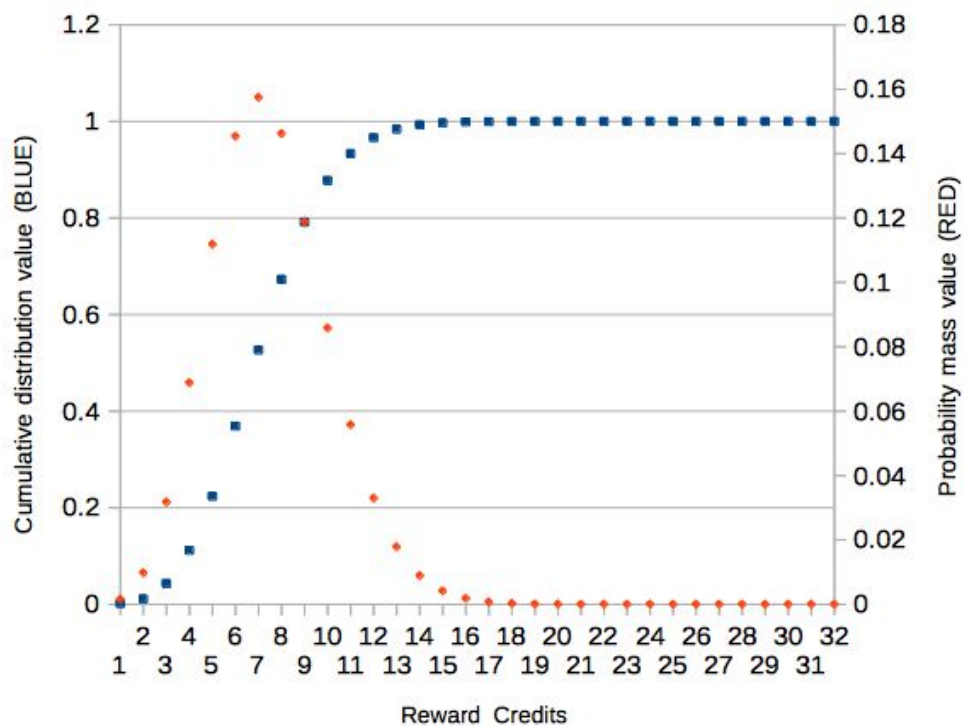
The result can be displayed as figure as well. As you can see, most of the approval rate from customers will be higher than before, and the cost of operating will be easily in control by operating department. The most important is, it is visible for everybody.

After optimization

average reward credits = 7.4999999



Cumulative distribution for program



Conclusion

Big idea will be very simple.

Behind this simple solution, it is a very confident, reasonable and verifiable mathematics principle.

This reward mechanism has been improved a lot by other colleagues later. Actually, continuous innovation in technology is the key for successful entrepreneurship to keep the advantage in intense competition.

Thanks for everyone.

10 July, 2016 @ Adelaide

References:

Poisson distribution

https://en.wikipedia.org/wiki/Poisson_distribution

LibreOffice

<https://www.libreoffice.org>