The Poisson Process: Meaning, Properties, Simulations

Giacomo Babudri

January 2024

1 Introduction

1.1 Definition

The Poisson Process is a stochastic process that models the number of events occurring in fixed intervals of time or space. Mathematically, it is defined by the probability mass function:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

A Poisson distribution is a discrete probability distribution, meaning that it gives the probability of a discrete (i.e., countable) outcome. For Poisson distributions, the discrete outcome is the number of times an event occurs, represented by k.

You can use a Poisson distribution to predict or explain the number of events occurring within a given interval of time or space. "Events" could be anything from disease cases to customer purchases to meteor strikes. The interval can be any specific amount of time or space, such as 10 days or 5 square inches.

In the Poisson process, the parameter λ (lambda) stands as a crucial metric, embodying the average rate of events occurring within a specified interval of time or space. Its interpretation is pivotal for comprehending the behavior and characteristics of the process in practical terms. The graph below shows examples of Poisson distributions with different values of λ.

1.2 Properties

The two properties that characterized a poisson process are:

- Individual events happen at random and independently. That is, the probability of one event doesn't affect the probability of another event.

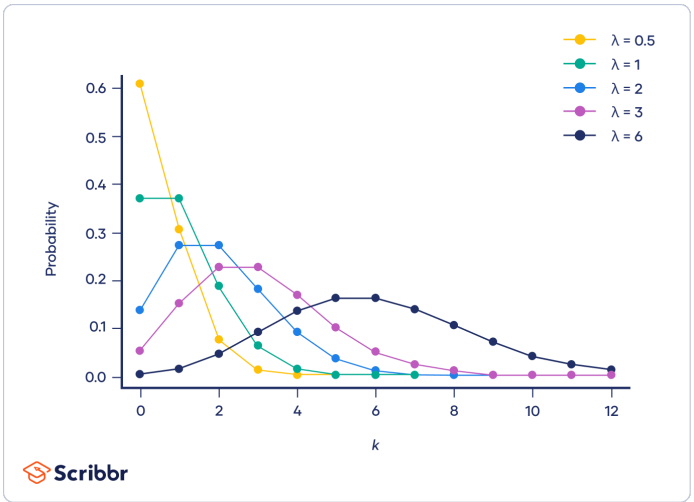

Figure 1: Poisson process Graph

- You know the mean number of events occurring within a given interval of time or space. This number is called λ (lambda), and it is assumed to be constant.

The properties of a Poisson process, namely the randomness and independence of individual events, as well as the constancy of the mean event rate represented by (lambda), play a crucial role in defining and characterizing this stochastic process. If these properties are violated, the system may deviate from the Poisson model. For instance, if events are not independent, the occurrence of one event could influence the likelihood of another, leading to a departure from the Poisson behavior. Recognizing and ensuring these properties is essential in accurately modeling and analyzing processes, as deviations may indicate different underlying stochastic behaviors.

2 Poisson Point Process

The Poisson Point Process stands as an extension and enrichment of the classical Poisson Process, introducing a broader framework that transcends temporal boundaries into the realm of spatial considerations. While the Poisson Process has been a cornerstone in modeling rare events occurring over time, the Poisson Point Process extends its applicability to scenarios where events are not only temporal but also spatially distributed. This chapter explores the nuances of the Poisson Point Process, aiming to provide a comprehensive understanding of its definition, properties, and applications.

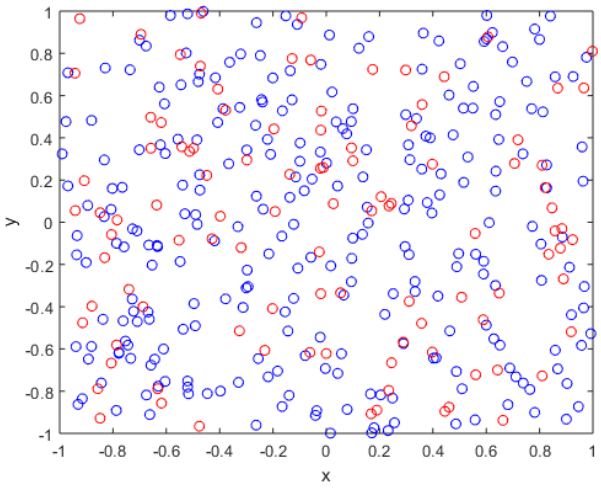

Figure 2: Example of poisson point process

2.1 Definition

A point process is a random collection of points falling in some space. In most applications, each point represents the time and/or location of an event. Poisson processes are a fundamental example of a point process that are used to model a variety of different phenomena and serve as the basis of more complicated processes. More in general in a Poisson point process, events are distributed randomly in space, and the probability of finding a certain number of events in a given region follows a Poisson distribution. The process is characterized by the Poisson parameter, often denoted as , which represents the average rate of events occurring in a unit area or volume.

2.2 Properties

1. **Independence of Events:** Like its temporal counterpart, the Poisson Point Process maintains the crucial property of independence. The occurrence of events at one point in space is entirely independent of events at other locations or times. This independence simplifies the modeling of complex systems and ensures that the past does not influence the future.

2. **Memorylessness:** The memoryless property persists in the spatial extension of the Poisson Process. The probability of an event occurring in a small spatial region remains constant and is not influenced by past events or the absence of events. This property greatly facilitates mathematical analysis and modeling.

3. **Intensity Function:** The process introduces the concept of an intensity

function, denoted by $\lambda(x, t)$, representing the rate at which events occur at a particular point in space (x) and time (t). The intensity function characterizes the spatial and temporal distribution of events, offering insights into the overall behavior of the process.

4. **Stationarity:** Stationarity implies that the statistical properties of the Poisson Point Process remain unchanged across different spatial and temporal regions. In other words, the process exhibits a consistent behavior, making it amenable to analysis and prediction.

5. **Spatial Homogeneity:** Spatial homogeneity indicates that the process's statistical properties are invariant across different regions of space. The likelihood of events occurring in one spatial region is the same as in any other region with equivalent characteristics. This property enhances the process's applicability to diverse spatial scenarios.

3 Applications

The Poisson process finds applications in various fields due to its ability to model the occurrence of rare and independent events.

3.1 Telecommunications

1. **Call Centers:** The Poisson process is used to model the arrival of phone calls at a call center. This helps in workforce management and resource allocation to handle varying call volumes efficiently.

2. **Computer Networks:** In computer networks, the Poisson process can model the arrival of data packets, aiding in the design and optimization of network protocols.

3.2 Economics and Finance

1. **Service Points:** Banks, supermarkets, and other service points often experience customer arrivals modeled by the Poisson process. This is essential for managing queues, staffing, and improving customer service.

2. **Stock Markets:** Financial transactions in stock markets can be modeled using a Poisson process, assisting in risk assessment, algorithmic trading, and market analysis.

3.3 Biology and Medicine

1. **DNA Mutations:** The distribution of mutations in a DNA sequence over time can be modeled using a Poisson process, aiding geneticists in understanding genetic variability.

2. **Rare Diseases:** Occurrence of rare diseases in a population, such as genetic disorders, can be studied using a Poisson process to assess prevalence and distribution.

3.4 Traffic Engineering

1. **Vehicle Arrivals:** Toll booths, traffic intersections, and highway segments can use the Poisson process to model the arrival of vehicles, facilitating traffic flow analysis and infrastructure planning.

2. **Accident Analysis:** The distribution of accidents or breakdowns on a road can be studied using a Poisson process to enhance road safety measures.

3.5 Insurance and Risk Management

1. **Insurance Claims:** The Poisson process is employed to model the arrival of insurance claims, assisting insurers in risk assessment, premium calculation, and resource allocation.

2. **Loss Distribution:** Analyzing the distribution of losses in a portfolio of assets helps in risk management strategies and financial planning.

4 simulation

4.1 Scenario

During the course, was executed a simulation, tryng to implement this scenario: "M servers are subject to attacks during a period of time T (for instance 1 year). Subdivide the interval T in N subinterval of size T/N and in each of this suppose that an attack can occur with probability T/N. Simulate the attacks to the M servers and represent each of them with a line which makes jumps of 1 at each attack event. Using the same objects ("movable/resizable rectangle", histogram, etc.) of the previous homework 3 draw vertically on the line chart the 2 histograms representing the distribution of the number of attacks at the end of the period and one internal istant for comparison."

4.2 Demonstration

To demonstrate if this simulation is a solid poisson process, it's important to study the two properties introduced and see if they are respected.

- The attacks are individual events, random, and, most importantly, independent from one another!

- There is a constant value λ, which we use to calculate the probability that an attack occurs ($\lambda \cdot \frac{T}{N}$). As we can see in the output of the code, particularly in the distribution, the value λ is also the mean of the distribution. Intuitively, we can see that the average for a fixed $\lambda = 50$ is, in fact, 50.

4.3 Output

It's interesting to compare the output of this simulation, after changing the value of N. As we know, the Law of Large Numbers (LLN) is a fundamental theorem in probability and statistics that describes the behavior of sample averages as the size of the sample increases.

4.3.1 Low value of N and M

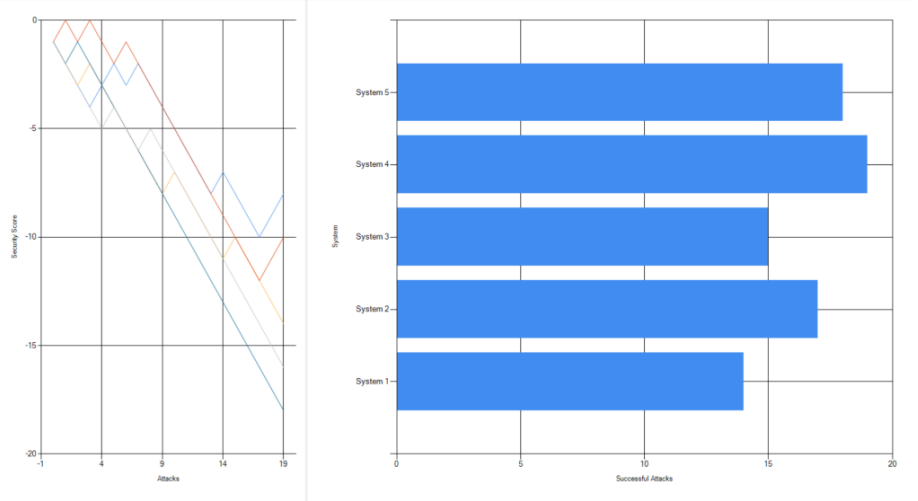

Figure 3: Simulation's output with low N

Considerations

With a low number of samples (N), the simulation becomes highly sensitive to the inherent randomness of the Poisson process. Since Poisson events are by definition rare and unpredictable, a small sample may not provide a representative snapshot of the process. The observed counts in a short time span may deviate significantly from the expected mean due to the stochastic nature of the process.

4.3.2 Large value N and sufficient large value M

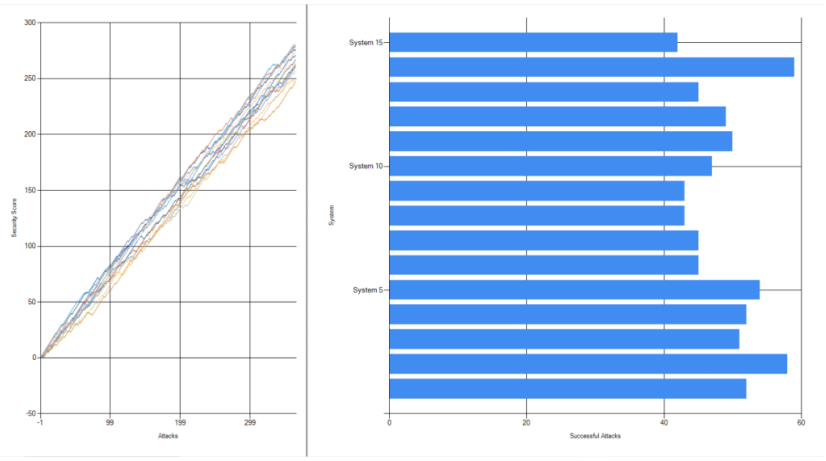

Figure 4: Simulation's output for large value N

Considerations

With a large number of samples (N), simulations of the Poisson process gain enhanced statistical power. The Law of Large Numbers (LLN) becomes more evident, allowing for more accurate estimation of parameters such as the mean rate of events. The observed counts tend to converge towards the expected mean, providing a robust foundation for statistical inference.

4.4 Code

The code was realized during the course, through the use of visual studio, and executed in c#.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
```

```
namespace Homeowrk {

    public partial class Form1 : Form
    {
        Chart chart1 = new Chart();
        Chart chart2 = new Chart();
        public Form1()
        {

            InitializeComponent();
            InitializeChart(chart1);
            InitializeChart(chart2);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Random random = new Random();
            int M = 15;
            int N =365;
            int T = 1;
            double lambda = 50;
            double p = lambda * ((double) T / N);
            p = 1 - p;

            int[][] Hystogram_matrix = AttacksLanded(M, N, p);

            CreateLineChartInt(M, N, Hystogram_matrix, chart1);
            CreateBarChartInt(M, N, Hystogram_matrix, chart2);

            var splitContainer = new SplitContainer();
            splitContainer.Dock = DockStyle.Fill;
            splitContainer.Panel1.Controls.Add(chart1);
            splitContainer.Panel2.Controls.Add(chart2);

            splitContainer.SplitterDistance = splitContainer.Height / 2;

            Controls.Add(splitContainer);
            this.Show();
        }

        private void CreateLineChartInt(int M, int N, int[][] AttackLanded, Chart chart)
        {
            for (int i = 0; i < M; i++)
            {
                Series series = new Series($"System {i + 1}");
```

```
            series.ChartType = SeriesChartType.Line;

            int conta = 0;
            for (int j = 0; j < N; j++)
            {
                conta += AttackLanded[i][j];
                series.Points.AddXY(j, conta);
            }

            chart.Series.Add(series);
        }

        chart.ChartAreas[0].AxisX.Title = "Attacks";
        chart.ChartAreas[0].AxisY.Title = "Security Score";
        chart.ChartAreas[0].RecalculateAxesScale();
    }

    private void CreateBarChartInt(int M, int N, int[][] AttackLanded, Chart chart)
    {
        Series barSeries = new Series("Successful Attacks");
        barSeries.ChartType = SeriesChartType.Bar;

        for (int i = 0; i < M; i++)
        {
            int successfulAttacks = 0;
            for (int j = 0; j < N; j++)
            {
                if (AttackLanded[i][j] == -1)
                {
                    successfulAttacks++;
                }
            }
            barSeries.Points.AddXY($"System {i + 1}", successfulAttacks);
        }
        chart.Series.Add(barSeries);

        chart.ChartAreas[0].AxisX.Title = "System";
        chart.ChartAreas[0].AxisY.Title = "Successful Attacks";
    }

    private int[][] AttacksLanded(int M, int N, double p)
    {

        Random random = new Random();
        int[][] Systems = new int[M][];
        for (int i = 0; i < M; i++)
```

```
            {
                int[] attacks = new int[N];
                for (int j = 0; j < N; j++)
                {

                    double att = random.NextDouble();
                    if (att < p)
                    {
                        attacks[j] = 1;
                    }
                    else
                    {
                        attacks[j] = -1;
                    }
                }
                Systems[i] = attacks;

            }
            return Systems;

        }

    }
}
```