Construct stochastic processes model to solve graphic and math problems

Shuyang Ding

College of Computer Science and Technology, Harbin Institute of Technology , Weihai, China

mofangmf@alumni.tongji.edu.cn

Abstract. The use of probabilistic reasoning has been used to a variety of applications, such as image identification, computer diagnostics, stock price prediction, movie recommendation, and cyber intrusion detection. However, the range of probabilistic programming was limited until recently (partly due to the low computing capacity), and the bulk of inference techniques needed to be developed manually for each task. However, in 2015, using a 50-line probabilistic computer vision program, 3D representations of human faces were produced from 2D images of those faces.Use a computer to model various complex stochastic phenomena. All projects are programmed in java language. Find the actual number of 1s in a binary number. Formulate and analyze mathematical models to real life phenomena like Data Matrix and QR code. Those code mentioned above can provide a link to the real-life website or some certain message needed to be shown. using some random numbers to model uncertainty. To model some fractals. The first fractal maybe not self-avoiding. But the second one will never meet the same path for twice. To apply the concept stochastic processes. More java features and some random fractals and regression. Multiple factor analysis.

Keywords: Stochasitic Process, Data Matrix and QR Code, Sierpinski Trangle, Fractals, Find 1s in a Binary Number.

1. Introduciton

Stochastic processes can be understood in this way. On a time axis, random experiments (which can be discrete or continuous) are continuously performed, and applied mathematician do not know the distribution that the results of each random experiment may obey, and the corresponding results at each time point. The distribution is unknown, that is, $X(t)$ which is unknown, and there are many cases. However, if applied mathematician start the experiment to a fixed end time point,applied mathematician can get a set of random variables $X(t)$, that is, each time point t,it corresponds to a random variable. Then, a family of (infinitely many) random variables corresponding to this series of it becomes a random process, denoted as $\{X(t), t \in T\}$.[1]Random walkers is a small part of the use of stochastic processes and so is the fractals.Random walker like some statistical model is full of uncertainty. They are constructed of random trajectories, each of which is random.

Random records formed by steps. The trajectories of molecular activities in gases or liquids can also be used as models for random walks. This concept was first proposed by Carl Pearson in 1905 and has been applied in many fields such as ecology, economics, psychology, computational science, physics, chemistry and biology. And Fractal, usually a rough or fragmented geometric shape, can be divided into several parts, and each part is a reduced shape of the whole. For example a kaleidoscope is a classic fractals. And this paper will use the knowledge from stochastic processes. To solve the promblem occurred in the face of uncertainty. Probabilistic programming give us help to make the conclusion. It make our congition of the environment with the laws of probability. There is a way making probabilistic systems much easier to construct. It is probabilistic programming which is a new way. Use a model under our domain and then make a decision. Conbined with some certain language and then applied mathematician can start to make a hypothesis. Take soccer as an example. It shows that around nine percent of the corner kicks will lead to the goal. To predict the result. Like the 6.4 veteran center forward playing the attacking team but the defender was just playing his first show[1-18].

To give it a ouve rall horizon, problem reasoning system is a smooth way to solve the uncertain problem. They can answer many problems under a certain situation.Always they can be divided in three branches. Predict future events. Infer the cause of events. To learn a better model.

2. In the random processes finding the 1s in a binary number

Somehow when applied mathematician generate a number into the very bottom of the computer. It transfers it into a binary number which only contains a combination of number 1 and 0. In military use of the decoding or hacking process.applied mathematician have to find out the 1s in the number soon as possible. The combination of 1s is a problem need to be solved especially when it comes to the long type of Int or String. The code is programmed in appendix 5.1.

- Give the code a package which it belongs to, the code is run in the package applied mathematician defined. When applied mathematician use some other outside functions, or some variables only can be found in the package, the code must be setteled in the same package so that applied mathematician could use them.
- mathematician have to write the function which can transfer the decimal number to binary number, First applied mathematician claim a temp which in the type of String, and claim the bin equals the result of the function of Interger to Binary(). After the function is done, the bin will get the result. Then claim a variable in the type of int. It equals to 12-bin.length();
- then take a judgment to bin.length(), if the length is a number under 12, make it into a 'for' loop, bin equals '0' plus bin.
- int one Num=0. For(int i=0;i<bin.length();i++), make the number at I 1.and one number equals itself plus one. If one Num==4 than just print the bin and the 1s it contains.The result is as below which is part of the whole result.

3. About the random walker

Model random walk in 2 and 3 dimensions. Random walks are useful models in many biological problems including folding of proteins and development of natural habitats of animals. At each discrete time step labeled by the variable i a "random walker" can make a step in random direction (left, right, back or forth in 2D, or left, right, back, forth, up or down in 3D). [4] The code computes the distance to the origin at all discrete instances of time and then plots these dependencies. The plots represent the distance a random walker made at a given time from the origin (that is, where the random walk has started). Note that in 2D a random walker will visit the origin again with probability 1; in 3D, the random walker will visit the origin again with a nonzero probability that is less than 1 (this probability is one of the random walk constants). This means that in 3D some "random walkers" will never return to the origin; in 2D all random walkers return to the origin sooner or later. 2 / 3 The main part of the homework is to find (estimate, fit to the data) the average distance ⟨r(t) and ⟩ ⟨r 2 (t) that a random walker has to the ⟩ origin after time. Note that time is discrete and represented by the variable i in the codes. The individual trajectories of random walkers will be very noisy and it will be difficult to find the above dependence.

The suggested approach is to generate many independent trajectories, say, labeled by j , and compute the average distance (and squared distance) to origin. When more and more trajectories get included in averaging, the resulting data set ⟨r(t) , will look increasingly smooth. ⟩ (a) Find the functions ⟨r(t) and ⟩ ⟨r 2 (t) numerically. ⟩ (b) Fit them to power functions using nonlinear regression (explained in the lectures) (c) Find the variance of the distance, ⟨r 2 (t) - ⟩ ⟨r(t)⟩ 2 I will show how to compute this quantity theoretically using central limit theorem. The code is as below. It's a random walker class, you have to new a Random () class to generate a random number. Int a two dimension array.

Which has a capacity of 200, and the 3 means there are three coordinates which refer to X,Y,Z. then int a flag, double the result. Double a t[], g[]. Double finn and fin. Write two 'for' loops.if flag equals to 1 then the random number is a positive one. Else it's a nagitive one. Let the former randomwalker plus the rear one. And r[n] equals to Math.pow((walker[m+1][n]-walker[m][n],2). The code is programmed in the appendix 5.2.

4. About the fractal

Fractal Theory is a very self-consistent and disciplined theroy. Fractal this name was first coined in 1975. Its original meaning is "irregular, Fractional, "fragmented" object, the noun is a reference to the Latin fractus (broken). It has the dual meaning of fraction (fraction), and it is a new geometry with irregular geometric forms as its research object. Benoit B. Mandelbrot's book "The Fractal Geometry of Nature" looks at the various forms of nature in a new geometric way.[4]

Beneath the surface of chaos lies a certain order, which can formulate coastlines, mountains, rivers, rocks, trees, forests, clouds, lightning, waves, and more. It is precisely because they have different formulas that they present different geometric patterns. In scientific research, modeling and analysis of many irregular objects, such as galaxy distribution, seepage, price fluctuations in financial markets and other complex objects, require a new geometry to describe. Therefore, "fractal" can generally be regarded as a state of aggregation of large and small fragments, which is a general term for graphics, structures and phenomena without characteristic lengths. The geometry describing fractals is called fractal geometry, also known as geometry describing nature. The code is programmed in the appendix 5.3.

mathematician write a trangle generating code, JF rame frame=JF rame() applied mathematician transfer the function. Set the frame.set Title and set Size of the fractal with width and height. Let frame add the former 'this'. Then applied mathematician can actually write the graphics.

5. About the data matrix

Data Matrix, formerly known as Data code, was invented in 1989. Data Matrix has two different forms like ECC000-140 and ECC200. ECC200 uses a well functioned algorithm to generate polynomials to calculate error correction codes. It needs to be printed in different sizes, but the error correction code used should match the size. Because its algorithm is easier and the size is more flexible, ECC200 is generally more common. [6] The code is programmed in appendix 5.4.

Before applied mathematician start to program,applied mathematician need to download a package from web. Zxing.jar from geogle is needed to generate a data matrix or qr code. To use the package,applied mathematician need to use Maven. The version of Maven must agree with the version of java. For example, jdk1.7 cooperate with Maven 3.6.1.

First applied mathematician have to claim that the backgroud of the data matrix. For example, the color is differed by black and white. And then write the main function which is the main part of our code.applied mathematician just start to try the code under. Giving it a flag of 518, applied mathematician start our way to generate the code, if the flag is not null, then applied mathematicia can assume that the pircture is successfully generated.

In the catch applied mathematician throw out the exceptions. If there is an error then means that applied mathematician fail to generate the data matrix. Then applied mathematician use the e.print Stack Trace ().

Mthematician write a function which has a boolean type of return. First applied mathematician define a website which the data matrix is leading to. Then applied mathematician define a path to store the picture generated. Then applied mathematician define the width and height of the picture. Then define the type of paper applied mathematician want to have.applied mathematician define the picture as jpg. Then applied mathematician setting the code to avoid the grabled code. Then set the parameters of the code and set the file.

At the end,applied mathematician set the image buffer. And give the picture a name.

6. Conclusion

In those four parts mentioned above, there are 4 different ways applied mathematician use the concept stochastic process, from some easy way to complicated way, to find 1s in the binary code is usually useful when applied mathematician decode some puzzles, or when applied mathematician input a number into some bottom layer computers which only have some poor computing power,applied mathematician have to make the number input very suitable for the way computer interact with the real world; Random walker, some random walker is very well-known, such as the drunken walker, Their steps follow some certain random track, if applied mathematician have an search on their law of their step, random walker can be a collection of multiple vectors. They are diversed in two way or three way; And fractals has a classic graphic like Sierpinski triangle. I put a square way to show how to compute the Sierpinski square; And about the data matrix, this code is generated by the package named Zxing from geogle, i set the environment of maven to cooperate with the version of jdk, and input the groupid and dependence of the jar package, then set the background and the color of the data matrix, the data matrix will lead a way to the exact website.

Appendix

Algorithm 1 Find the 1s in the binary number

```
package A1;
import java.util.Scanner;
public class TOTALNUM {
   public static void main(String[] args) {
      for (int num = 0; num <= 4096; num++) {
         binOneNum(num);
      }
   }
   public static void binOneNum(int n) {
      String temp="";
       String bin = Integer.toBinaryString(n);
      int bit=12-bin.length();
      if(bin.length()<12) {
         for (int i = 0; i < bit; i++) {
            bin = "0" + bin;
         }
      }
       int oneNum = 0;
       for (int i = 0; i < bin.length(); i++) {
          if (bin.charAt(i) == '1') {
             oneNum++;
          }
       }
       if(oneNum==4) {
          System.out.println(bin);
          System.out.println("this contains 4 1s"+oneNum);
       }
   }
}
```

Algorithm 2 Random walker

```
package DO;
import java.util.Random;
public class randomwalker {
  public static void main(String[] args) {
    Random c = new Random();
    Random d = new Random();
    int[][] walker = new int[200][3];
    walker[0][0] = 0;
    walker[0][1] = 0;
    walker[0][2] = 0;
    double[] r = new double[3];
    int flag;
    double result=0;
    double t[] = new double[200];
    double g[]=new double[200];
    double fin=0;
    double finn=0;
      for (int m = 0; m <= 100; m++) {
      for (int n = 0; n <= 2; n++) {
        flag = d.nextInt(1);
        if (flag == 0) {
          walker[m+1][n] = walker[m][n] + c.nextInt(10);
          r[n] = Math.pow((walker[m+1][n] - walker[m][n]), 2);
        } else {
          walker[m+1][n] = walker[m][n] - c.nextInt(10);
          r[n] = Math.pow((walker[m+1][n] - walker[m][n]), 2);
        }
      }
      for (int j = 0; j <=2 ; j++) {
        result = result + r[j];
      }
        t[m]=result;
        g[m]=Math.pow(result,0.5);
        result=0;
      }
    for (int i = 0; i <100 ; i++) {
      fin=fin+t[i];
      finn=finn+g[i];
    }
     fin=fin/100;
     finn=finn/100;
    System.out.println("the r^2(t) equals="+fin);
    System.out.println("the r(t) equals="+finn);
    }
}
```

Algorithm 3 Fractal

```
package homework;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class Tangle extends JPanel{
   public void show(){
      JFrame frame=new JFrame();
      frame.setTitle("square fractal");
      frame.setSize(600,600);
      //frame.setResizable(false);
      frame.setLocationRelativeTo(null);
      frame.setDefaultCloseOperation(3);
      this.setBackground(Color.WHITE);
      frame.add(this);
      frame.setVisible(true);
      draw(this.getGraphics());

   }
   public void paint(Graphics g){
      super.paint(g);
      draw(g);
   }
   public void draw(Graphics g){
      tangle(5,g,200,200,100,100);
   }
   public void tangle(int count,Graphics g,int x1,int y1,int width,int height){
      if(count==0){
         return;
      }
      g.fillRect(x1,y1, width, height);
tangle(count-1,g,(int)(x1+(1.0/3)*width),(int)(y1-(2.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1+(1.0/3)*width),(int)(y1+(4.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1-(2.0/3)*width),(int)(y1+(1.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1+(4.0/3)*width),(int)(y1+(1.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1-(2.0/3)*width),(int)(y1-(2.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1+(4.0/3)*width),(int)(y1-(2.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1-(2.0/3)*width),(int)(y1+(4.0/3)*height),width/3,height/3);
tangle(count-1,g,(int)(x1+(4.0/3)*width),(int)(y1+(4.0/3)*height),width/3,height/3);
   }
   public static void main(String[] args) {
      Tangle tan=new Tangle();
      tan.show();
   }
}
```

Algorithm 4 Data Matrix

```
package matrix;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Hashtable;
import java.nio.file.Path;
import java.util.HashMap;
import java.util.Map;
import javax.imageio.ImageIO;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.datamatrix.DataMatrixWriter
public class data matrix {
   private static final int BLACK = 0xFF000000;
   private static final int WHITE = 0xFFFFFFFF;
   public static void main(String[] args) {
      try {
         boolean flag = generateCode("518");
         if (flag) {
            System.out.println("successfully generated");
         }
      } catch (WriterException | IOException e) {
         System.err.println("fail to generate");
         e.printStackTrace();
      }
   }
   public static boolean generate Code(String productId) throws Writer Exception, IOException {
      String text = "https://www.4399.com/?uudi=" + productId;
      String path = "C:/soft";
     BarcodeFormat format = BarcodeFormat.DATA_MATRIX;
      int width = 2000;
      int height = 2000;
      String mat = "jpg";
      final Map<EncodeHintType,Object> hints = new HashMap<>();
      hints.put(EncodeHintType.CHARACTER_SET, "utf8");
      BitMatrix bitMatrix = new  DataMatrixWriter().encode(text,BarcodeFormat.DATA_MATRIX, width,
height, hints);
      File outputFile = new File(path);
      if (!outputFile.exists()) {
         outputFile.mkdir();
      }
      int b_width = bitMatrix.getWidth();
      int b_height = bitMatrix.getHeight();
      BufferedImage image = new BufferedImage(b_width, b_height, BufferedImage.TYPE_3BYTE_BGR);

      for (int x = 0; x < b_width; x++) {
         for (int y = 0; y < b_height; y++) {
            image.setRGB(x, y, bitMatrix.get(x, y) ? BLACK : WHITE);
         }
      }
      ImageIO.write(image, "jpg", new File(path + "/code." + mat));
      return true;
   }
}
```

References

[1] Avi Pfeffer. Practical Probabilistic Programming.
[2] Rick Durrett. Probability Theory and Examples. Fourth Edition.
[3] ZouYang, WuHecheng, Zhao Yingding, Jiang Yunzhi. Random walk recommendation algorithm based on multi-weight similarity [J]. Computer Application Research,

2020,37(11):3267-3270+3296.DOI:10.19734/j.issn. 1001-3695.2019.08.0275.

[4] Zou [1] Liao Yongxin. Research on the Joint Classification Algorithm of Heterogeneous Label Sets Based on Random Walk and Dynamic Label Propagation [D]. South China University of Technology, 2017.

[5] Lu Yuke. Research on Complex Code Recognition Technology [D]. University of Electronic Science and Technology of China, 2022. DOI: 10.27005/d.cnki.gdzku.2022.003580.

[6] Song Bin, Liu Lili, Zhang Lei, Wang Lei, Du Yuxin, Zhang Ning. Information Management of Coal Mine Equipment Based on Data Matrix Code [J]. Industrial and Mining Automation, 2020, 46(11): 83-86+94. DOI: 10.13272/ j.issn.1671-251x.2020050059.

[7] Yuan Tailing, Xu Kun. Length minimization algorithm of two-dimensional code bit stream [J]. Chinese Journal of Image and Graphics, 2022, 27(08): 2356-2367.

[8] GB/T 41208-2021, Data Matrix Code [S].

[9] Deng Huipeng, Wang Yi, Dong Xiaowen. Comparison of Hanxin code with PDF417, data matrix code (DM code), and QR code [J]. China Automatic Identification Technology, 2021(05):50-53.

[10] Guo Lijie. Random walk and its application in leaf image segmentation [D]. Lanzhou University, 2018.

[11] [Karatzas, I. and S. E. Shreve (1991). Brownian Motion and Stochastic Calculus. Springer-Verlag, New York.

[12] Lukacs, E. (1970). Characteristic Functions, 2nd Ed. Griffin.

[13] Meyn, S. P. and Tweedie, R. L. (1993). Markov Chains and Stochastic Stability. Springer-Verlag, New York.

[14] Petrov, V. V. (1995). Limit Theorems of Probability Theory. Oxford University Press, Oxford.

[15] Samorodnitsky, G. and M. S. Taqqu. (1994). Stable Non-Gaussian Random Processes. Chapman-Hall/CRC,

[16] Shorack, G. R. (2000). Probability for Statisticians. Springer-Verlag, New York.

[17] Durrett, R. (2019). Probability: theory and examples (Vol. 49). Cambridge university press.

[18] Chung, K. L. (1967). Markov Chains with Stationary Transition Probabilities, 2nd Ed. Springer-Verlag.