**宋运翔 计算金融 2016141223037**

1.采用数学归纳法，假设当时成立，。之后证明当时也成立。

由



且已知该期权依赖为非路径依赖，因此参数最高为一次项



且由于



因此可以得到。

2.编写程序时我们的核心公式为和，其中可以发现等式左边为离散型，右边则为连续型，用离散型逼近得到离散型结果，且当越大，两者的结果更为接近，离散化则是为了方便使用。

3.编写程序时，我采用JAVA进行实现，算法流程如下：



其中叶子结点向量即每条路径的最后的股票价格，期权向量指该路径对应的期权价格，通过计算得出，之后带入期望求解公式得到：



由于JAVA绘图并不美观，之后导出数据，使用MATLAB绘制图像如下：

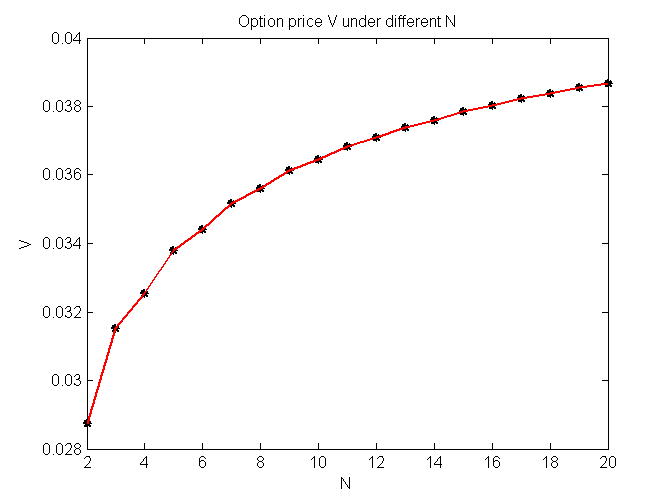


图3.1不同N时的期权价格V

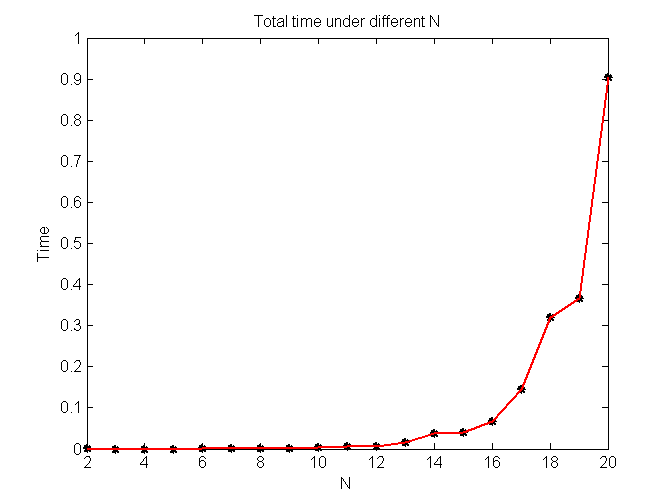


图3.2不同N时的运算时间Time

4.假设持有回望期权多头，0时刻持有资本，期望在t时刻获得收益为：



通过复制期权计算，0时刻买入股票对冲空头：



花费，剩余投资到无风险利率为r的货币市场。因此，无论股票上涨还是下跌，在t时刻都能获得持有价值为的资产组合。

附录

|  |
| --- |
|  |
|  | import java.util.\*;  import java.lang.\*; |
|  | import java.io.\*; |
|  | class Node{ |
|  | public double data; //节点数据 |
|  | Node leftChild; //左子节点的引用 |
|  | Node rightChild; //右子节点的引用 |
|  | } |
|  |  |
|  | class BinaryTree { |
|  | Node root; |
|  |  |
|  | public BinaryTree() { |
|  | this.root = null; |
|  | } |
|  |  |
|  | public static Node BuildTree(Node node, double S, double u, double d, int N) { |
|  | node = new Node(); |
|  | node.data = S; |
|  | //根据步数建立期权二叉树 |
|  | if (N > 0) { |
|  | node.leftChild = BuildTree(node.leftChild, S \* u, u, d, N - 1); |
|  | node.rightChild = BuildTree(node.rightChild, S \* d, u, d, N - 1); |
|  | } |
|  | return node; |
|  | } |
|  | } |
|  |  |
|  | class findallpaths{ |
|  |  |
|  | //每条路径叶子结点向量 |
|  | public ArrayList<Double>Endcolumn= new ArrayList<>(); |
|  | //每条路径期权向量 |
|  | public ArrayList<Double>Vcolumn=new ArrayList<>(); |
|  |  |
|  | public void allpaths(Node root) { |
|  |  |
|  | //储存遍历的路径 |
|  | ArrayList<Double> arrayListSingle = new ArrayList<>(); |
|  | //先存入根节点 |
|  | arrayListSingle.add(root.data); |
|  | traverse(root,arrayListSingle); |
|  | } |
|  |  |
|  | //前序递归遍历 |
|  | public void traverse(Node node,ArrayList<Double>arr){ |
|  | //遍历左节点 |
|  | if(node.leftChild != null) { |
|  | //在遍历的路径中添加马上开始遍历的左节点,开始遍历左节点 |
|  | arr.add(node.leftChild.data); |
|  | traverse(node.leftChild,arr); |
|  | //遍历完左节点,从遍历的路径中删除左节点 |
|  | arr.remove(arr.size() - 1); |
|  | } |
|  | //遍历右节点 |
|  | if(node.rightChild != null){ |
|  | //在遍历的路径中添加马上开始遍历的右节点,开始遍历右节点 |
|  | arr.add(node.rightChild.data); |
|  | traverse(node.rightChild,arr); |
|  | arr.remove(arr.size() - 1); |
|  | } |
|  | //当左右节点都空时,说明这个节点是叶子节点 |
|  | if(node.leftChild == null && node.rightChild == null) { |
|  | Endcolumn.add(arr.get(arr.size()-1)); |
|  | Vcolumn.add(Collections.max(arr)-arr.get(arr.size()-1)); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | class TreeApp { |
|  |  |
|  | public static void main(String[] args) { |
|  | //输入参数为现价S0，连续利率R，连续标准差sigma，总步数N，时间跨度t |
|  | double S0 = 1, R = 0.05, sigma = 0.2, t = 0.0833; |
|  | int N; |
|  | //定义输出字符串向量 |
|  | ArrayList<String>plotdata=new ArrayList<>(); |
|  | for(N=2;N<=20;N++){ |
|  | //二叉树参数 |
|  | double TimeA=System.currentTimeMillis(); |
|  | double sigmaN = Math.sqrt((sigma \* sigma \* t) / N); |
|  | double rN = Math.pow((1 + R), (t / N)) - 1; |
|  | double u = Math.exp(sigmaN); |
|  | double d = 1 / u; |
|  | double p = (1 + rN - d) / (u - d); |
|  |  |
|  | //建立期权二叉树 |
|  | BinaryTree tree = new BinaryTree(); |
|  | tree.root = BinaryTree.BuildTree(tree.root, S0, u, d, N); |
|  | //获得每条路径的期权向量Vcolumn和叶子结点向量Endcolumn |
|  | findallpaths find=new findallpaths(); |
|  | find.allpaths(tree.root); |
|  | //计算期权价格 |
|  | //初始化期权期望价格为V0 |
|  | double V=0; |
|  | for (int i=0;i<find.Vcolumn.size();i++){ |
|  | /\*M为下跌的次数，因此N-M为上涨的次数 |
|  | 计算方式为S0\*u^(N-M)\*d^M=Endcolumn[i] |
|  | 两边取对数，化简计算式为(Nlnu-lnEndcolumn[i]+lnS0)/lnu-lnd; |
|  | \* \*/ |
|  | int M=(int)Math.round((N\*Math.log(u)-Math.log(find.Endcolumn.get(i))+Math.log(S0))/(Math.log(u)-Math.log(d))); |
|  | //每条路径期权为Vcolumn[I]=p^(N-M)+(1-P)^M |
|  | V+=find.Vcolumn.get(i)\*Math.pow(p,(N-M))\*Math.pow((1-p),M); |
|  | } |
|  | //期权价格V0,利用期望求解法，V0=E[V]/(1+rN)^N |
|  | double V0=V/Math.pow((1+rN),N); |
|  | double TimeB=System.currentTimeMillis(); |
|  | double Time=(TimeB-TimeA)/1000; |
|  | //输出结果 |
|  | System.out.println("N="+N+"时，期权价格为："+V0+",运算时间为："+Time+"s。"); |
|  | //将结果添加到输出序列 |
|  | plotdata.add(N+" "+V0+" "+Time); |
|  |  |
|  | } |
|  | //考虑到java绘图并不美观，将计算数据输出到文本"plotdata.txt"中，使用matlab绘图 |
|  | FileWriter fileWriter = null; |
|  | try { |
|  | fileWriter = new FileWriter("D:/Documents/GitHub/First/Financial-stochastic-analysis/plotdata.txt");//创建文本文件 |
|  | int i=0; |
|  | while(i<19){//循环写入 |
|  | fileWriter.write(plotdata.get(i)+"\r\n");//写入 \r\n换行 |
|  | i++; |
|  | } |
|  | fileWriter.flush(); |
|  | fileWriter.close(); |
|  | } catch (IOException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | } |
|  |  |