3 树形计算回望期权价格

使用申明*

2021 年 2 月 20 日

目录

1 简介

1.1 回望期权

回望期权是种价格依赖于股票历史价格极值的期权。如果回望期权为只能在到期日执行的欧式期权，其价格有解析解。不过这里我们主要使用二叉树法对回望期权当前价格进行数值计算。

对于浮动回望看涨期权，其到期日时价值为 $\max(S_T - S_{min}, 0)$，其中 S_T 为执行时间 T 时股票价格，S_{min} 为有效期内股票历史最低价格。类似地，浮动回望看跌期权到期时价值为 $\max(S_{max} - S_T, 0)$，其中 S_{max} 为期权有效期内股票的历史最高价格。

1.2 回望期权分类

1. 浮动/固定回望期权：浮动回望期权的收益如上文所述。固定回望期权和普通期权相似，对于固定回望看涨期权，其收益为期限内股票最高价格超出约定执行价格的差价，对于固定回望看跌期权，收益为约定执行价格大于期限内股票最低价格的价差。

2. 欧式/美式回望期权：无论浮动或是固定回望期权，都可以是欧式或美式。如果期权为美式期权，则持有者有权选择在有效期内任意时刻执行期权。

2 数值计算步骤

以欧式浮动回望看涨期权为例，我们简单描述一下计算过程。首先还是建立一个股票价格变化的树形。对于普通欧式期权的二叉树定价，我们知道在树形每个节点，节点上的股票价格都对应于一个期权价格。但是在回望看涨期权的情况下，不是每个股票价格都对应于一个期权价格，而是节点上每个可能的历史最低价格对应于一个期权价格，所以每个节点需要计算和保存在该处所有可能的股票历史最低价格。然后计算出树形末端执行时刻期权在每个节点每个历史最低股票价格对应的价格，往回递推出根节点处回望期权的价格。递推计算是用每个节点处每个历史最低价格，可能到达的下一层的节点的对应历史最低价格所对应的期权价格，进行加权平均和贴现。

具体步骤为：

1. 根据给定的相关参数计算出二叉树分叉参数 p, u, d。并建立股票价格变化树形。

$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = 1/u, \quad p = \frac{e^{r\Delta t} - d}{u - d}. \tag{1}$$

2. 从根节点处开始一层一层计算，在每层每个节点处计算出所有可能的历史最低股票价格并保存下来。

3. 在树形叶子层计算出每个节点每个可能的历史最低股票价格所对应的期权价格并保存下来。

4. 往上一层递推，计算出该层每个节点的每个可能的历史最低股票价格对应的期权价格。具体在每个节点，其每个可能的历史最低股票价格都对应于一类历史路径，在该类历史路径情况下，股票在下一步价格上升或下降都分别对应于下一层一个节点的一个历史最低股票价格。所以该节点的一个历史最低股票价格对应的期权价格，可以通过对股票上升或下降后到达的下一层的历史最低股票价格所对应的期权价格进行加权平均并贴现。

5. 重复过程 4，直到计算出根节点处期权的价格。

对于美式回望看涨期权，在过程 4 中需要考虑在每个节点的每个历史最低股价的情景下应不应该执行期权，其它部分的过程不变。

3 步骤 Python 代码实现

```
import numpy as np

E = np.e

class Tree_lookback_option:
    def __init__(self, r, sigma, S_0, T, steps, strike_price=None, option_type="european"):
        """ 初始化实例。
        """
        self.r = r
        self.sigma = sigma
        self.S_0 = S_0
        self.T = T
        self.steps = steps
        self.option_type = option_type
        self.dt = self.T/self.steps
        self.u = E**(self.sigma*self.dt**0.5)
        self.d = 1/self.u
        self.p = (E**(self.r*self.dt)- self.d)/(self.u- self.d)
        # 如果strike_price为None，则期权为浮动回望期权，否则期权为固定回望期权（执行价格为strike_price)。
        self.strike_price = strike_price
        self.call_price = None
        self.put_price = None
```

```
        # 构建一个树形。
        self.tree = list()
        self.build_tree()

    def build_tree(self):
        """ 构建股票价格变化树形，并计算出每个节点所包含的所有可能的历史股票价格的极值。
        """
        # 简化参数名称。
        S_0, steps = self.S_0, self.steps
        u, d, p = self.u, self.d, self.p
        self.tree = list()
        for lvl in range(steps+1):
            row = list()
            for j in range(lvl+1):
                node = dict()
                node["S"] = S_0*(u**j)*(d**(lvl-j))
                node["Smin_vals"] = dict()
                node["Smax_vals"] = dict()
                # 确定每个节点可能有的历史最大和最小值，并把对应期权价格初始化为None。
                mins = lvl-j-max(0, lvl-2*j)+1
                maxs = j-max(0, 2*j-lvl)+1
                for k in range(mins):
                    node["Smin_vals"][lvl-j-k] = None
                for k in range(maxs):
                    node["Smax_vals"][-j+k] = None
                row.append(node)
            self.tree.append(row)
        return

    def calculate_call_price(self):
        """ 计算回望看涨期权价格。
        """
        r, S_0, steps = self.r, self.S_0, self.steps
        dt, u, d, p = self.dt, self.u, self.d, self.p
        strike_price = self.strike_price
        # 每步的贴现系数。
        a = E**(-r*dt)
        # 计算出期权在树形末端的价格。
        for node in self.tree[-1]:
            if strike_price is None:
                for d_num in node["Smin_vals"]:
                    node["Smin_vals"][d_num] = max(0, node["S"]-S_0*(d**d_num))
            else:
                for d_num in node["Smax_vals"]:
                    node["Smax_vals"][d_num] = max(0, S_0*(d**d_num)-strike_price)
        # 将期权价格递推回根节点。
        for lvl in range(steps-1, -1, -1):
            for j in range(lvl+1):
                node = self.tree[lvl][j]
                if strike_price is None:
                    for d_num in node["Smin_vals"]:
                        node["Smin_vals"][d_num] = p*a*self.tree[lvl+1][j+1]["Smin_vals"][d_num]
```

```
                    node["Smin_vals"][d_num] += (1-p)*a*self.tree[lvl+1][j]["Smin_vals"]\
                                [max(d_num, lvl-2*j+1)]
                    if self.option_type != "european":
                        node["Smin_vals"][d_num] = max(node["Smin_vals"][d_num], \
                                    node["S"]-S_0*(d**d_num))
                else:
                    for d_num in node["Smax_vals"]:
                        node["Smax_vals"][d_num] = (1-p)*a*self.tree[lvl+1][j]["Smax_vals"][d_num]
                        node["Smax_vals"][d_num] += p*a*self.tree[lvl+1][j+1]["Smax_vals"]\
                                [min(d_num, lvl-2*j-1)]
                        if self.option_type != "european":
                            node["Smax_vals"][d_num] = max(node["Smax_vals"][d_num], \
                                    S_0*(d**d_num)-strike_price)

        if strike_price is None:
            self.call_price = self.tree[0][0]["Smin_vals"][0]
        else:
            self.call_price = self.tree[0][0]["Smax_vals"][0]
        return

    def calculate_put_price(self):
        """ 计算回望看跌期权价格。
        """
        r, S_0, steps = self.r, self.S_0, self.steps
        u, d, p = self.u, self.d, self.p
        dt, strike_price = self.dt, self.strike_price
        a = E**(-r*dt)
        for node in self.tree[-1]:
            if strike_price is None:
                for d_num in node["Smax_vals"]:
                    node["Smax_vals"][d_num] = max(0, S_0*d**d_num-node["S"])
            else:
                for d_num in node["Smin_vals"]:
                    node["Smin_vals"][d_num] = max(0, strike_price-S_0*d**d_num)
        for lvl in range(steps-1, -1, -1):
            for j in range(lvl+1):
                node = self.tree[lvl][j]
                if strike_price is None:
                    for d_num in node["Smax_vals"]:
                        node["Smax_vals"][d_num] = (1-p)*a*self.tree[lvl+1][j]["Smax_vals"][d_num]
                        node["Smax_vals"][d_num] += p*a*self.tree[lvl+1][j+1]["Smax_vals"]\
                                [min(d_num, lvl-2*j-1)]
                        if self.option_type != "european":
                            node["Smax_vals"][d_num] = max(node["Smax_vals"][d_num], \
                                    S_0*(d**d_num)-node["S"])
                else:
                    for d_num in node["Smin_vals"]:
                        node["Smin_vals"][d_num] = p*a*self.tree[lvl+1][j+1]["Smin_vals"][d_num]
                        node["Smin_vals"][d_num] += (1-p)*a*self.tree[lvl+1][j]["Smin_vals"]\
                                [max(d_num, lvl-2*j+1)]
                        if self.option_type != "european":
                            node["Smin_vals"][d_num] = max(node["Smin_vals"][d_num], \
```

```
                strike_price-S_0*(d**d_num))
        if strike_price is None:
            self.put_price = self.tree[0][0]["Smax_vals"][0]
        else:
            self.put_price = self.tree[0][0]["Smin_vals"][0]
        return
```

4 计算示例

考虑当无风险利率为 0.1，波动率为 0.4，股票初始价格为 50，执行时间为 3 个月后的几种回望期权。使用步数为 5 步的二叉树计算期权价格。对于固定回望期权我们选取执行价格为 49。计算结果如下：

```
print("r=0.1, sigma=0.4, S_0=50, T=0.25, steps=5 ")
tree_obj = Tree_lookback_option(0.1, 0.4, 50, 0.25, 5)
tree_obj.calculate_call_price()
tree_obj.calculate_put_price()
print("欧式浮动看涨: {0:.5f}".format(tree_obj.call_price))
print("欧式浮动看跌: {0:.5f}".format(tree_obj.put_price))

tree_obj = Tree_lookback_option(0.1, 0.4, 50, 0.25, 5, option_type="american")
tree_obj.calculate_call_price()
tree_obj.calculate_put_price()
print("美式浮动看涨: {0:.5f}".format(tree_obj.call_price))
print("美式浮动看跌: {0:.5f}".format(tree_obj.put_price))

print("\n r=0.1, sigma=0.4, S_0=50, T=0.25, steps=5, strike_price=49 .")
tree_obj = Tree_lookback_option(0.1, 0.4, 50, 0.25, 5, strike_price=49)
tree_obj.calculate_call_price()
tree_obj.calculate_put_price()
print("欧式固定看涨: {0:.5f}".format(tree_obj.call_price))
print("欧式固定看跌: {0:.5f}".format(tree_obj.put_price))

tree_obj = Tree_lookback_option(0.1, 0.4, 50, 0.25, 5, strike_price=49, option_type="american")
tree_obj.calculate_call_price()
tree_obj.calculate_put_price()
print("美式固定看涨: {0:.5f}".format(tree_obj.call_price))
print("美式固定看跌: {0:.5f}".format(tree_obj.put_price))

Output:
r=0.1, sigma=0.4, S_0=50, T=0.25, steps=5.
欧式浮动看涨: 6.48347
欧式浮动看跌: 5.69116
美式浮动看涨: 6.48347
美式浮动看跌: 5.91857

r=0.1, sigma=0.4, S_0=50, T=0.25, steps=5, strike_price=49 .
欧式固定看涨: 7.90097
欧式固定看跌: 4.58603
美式固定看涨: 7.92152
美式固定看跌: 4.59751
```

5 相关说明

5.1 节点历史股票价格极值的计算

在描述股价变化过程的二叉树上，考虑任一节点，从树形根节点到达该节点都有一些可能路径，每一条可能的路径都有一个路径上最高和最低股价。到达一个节点的所有可能路径的历史最低/最高股价的可能取值，为该节点的所有可能历史股价极值。如果层数计数 i 从 0 开始，节点计数 j 也从 0 开始（由下往上）。对于在第 i 层的节点，每个节点位置的股价都经历了总计 i 次上升或下降变化，且上升比例 u 和下降比例 d 有关系 $ud = 1$。

如果我们想确定第 i 层，第 j 个节点的所有可能历史最低股价。不考虑次序，第 i 层第 j 个节点一共经历了 j 次上升，$(i-j)$ 次下降。达到该节点的路径的历史最低价格的最小值为 $S_0 d^{i-j}$，对应于一条先下降 $(i-j)$ 次，后上升 j 次的路径。且到达该节点的路径的历史最低价格的最大值为 $\min(S_0, S_0 u^j d^{i-j}) = \min(S_0, S_0 d^{i-2j})$，为路径起点和终点位置股价的较小值。所以路径历史最低价格的取值范围为 $[S_0 d^{i-j}, \min(S_0, S_0 d^{i-2j})]$，由于在树形上股价的可能取值为离散的 $S_0 d^k$，k 为整数，且对于 $S_0 d^{i-j}, S_0 d^{i-j-1}, ..., \min(S_0, S_0 d^{i-2j})$ 中每个值，我们都可以在树形上找到一条路径连接根节点和考虑节点，使得该路径上的历史最低价格为此值。所以对于第 i 层，第 j 个节点，该节点上所有可能的历史最低股价为 $S_0 d^{i-j}, S_0 d^{i-j-1}, ..., \min(S_0, S_0 d^{i-2j})$。

类似地，如果想确定第 i 层，第 j 个节点的所有可能历史最高股价。不考虑次序，到达该节点一共经历了 j 次上升，$(i-j)$ 次下降。到达该节点的路径的历史最高价格的最小值为 $\max(S_0, S_0 d^{i-2j})$，为路径起点和终点位置股价的较大值。对于第 i 层，第 j 个节点，该节点上所有可能的历史最高股价为 $\max(S_0, S_0 d^{i-2j}), ..., S_0 d^{1-j}, S_0 d^{-j}$。

由上面讨论，我们注意到对于树形上任一节点，到该节点的路径的历史股价极值的所有可能值可以用 $S_0 d^k$ 表示，k 为在一个范围内连续变化的整数。所以在程序实现中，可以用 dictionary 来存储股价极值和对应期权价格，key 为股价极值 $S_0 d^k$ 中的 k，value 为该极值下期权价格。

5.2 期权价格的递推

我们以美式浮动回望看涨期权为例。该期权在执行日期时的价格为即期股价大于股票历史最低价格的值。在构建好股价变化的树形并计算出每个节点对应的所有可能历史股价极值之后。我们可以首先计算出执行日期时期权的价格，即树形上叶子层的期权价格。然后往根节点处逐层递推。

对于浮动回望看涨期权，我们需要考虑的历史极值为股价历史最小值。假设我们在树形上第 i 层，第 j 个节点，在计算该节点处股价历史最小值 $S_0 d^k$ 所对应的期权的价格的时候。该处期权价格为下一步可能状态对应的期权价格的期望的贴现。有 p 的概率，一步后会到第 $i+1$ 层，第 $j+1$ 个节点，由于股价上升了，所以到达的该节点的股价历史最小值和上一步处的历史最小值相同，同为 $S_0 d^k$。有 $(1-p)$ 的概率，一步后会到第 $i+1$ 层，第 j 个节点，股价历史最小值变为 $\min(S_0 d^k, S_0 d^{i+1-2j})$，因为有可能这一步的下降使得股价最小值变得更低了。对这两种情况下对应的期权价格进行加权求和，然后乘以一步的贴现因子，即得到我们考虑的节点的历史股价极值对应的期权价格。在美式期权的情况下，要进一步将所得价格和该节点该历史股票极值的状态下执行期权的收益进行比较，最终期权价格取较大值。以此方式计算每一层每一个节点每一个历史极小股价对应的期权价格，直到根节点处。

对于其它几种回望期权，方式类似，需要考虑到达节点路径的历史股价极大值，还是极小值，由期权具体类型决定。

6 参考资料

参考文献

[1]《期权、期货及其他衍生产品》（原书第 9 版）第 27 章，John C. Hull 著，王勇、索吾林译，机械工业出版社，2014.11 。