

## 2 二叉树计算欧式和美式期权价格

使用申明\*

2021 年 2 月 18 日

### 目录

1 简介	1
2 二叉树计算期权价格步骤	2
3 步骤 Python 代码实现	2
4 计算示例	4
5 参考资料	4

### 1 简介

当期权为股票期权，二叉树是指股票价格在期限内可能的变化路径图形。考虑时间段为 0 至  $T$ ，步数为  $N$  的二叉树，在  $t = 0, \dots, (N-1)\Delta t$  的时间节点上股票价格有概率  $p$  由当前价格  $S_t$  变为  $uS_t$ ，有概率  $(1-p)$  变为  $dS_t$ 。其中  $\Delta t = \frac{T}{N}$ ， $u$  和  $d$  分别为上升和下降幅度。当二叉树的步数足够多时，股票价格的最后分布将为对数正态分布。

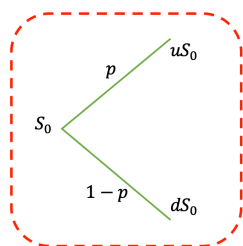


Fig. 1

$$\begin{aligned}u &= e^{\sigma\sqrt{\Delta t}} \\d &= e^{-\sigma\sqrt{\Delta t}} \\p &= \frac{e^{r\Delta t} - d}{u - d}\end{aligned}$$

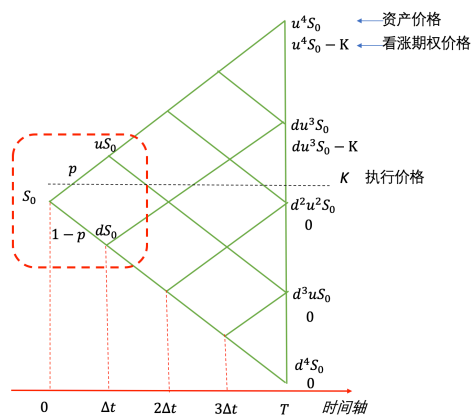


Fig. 2

《期权、期货及其他衍生产品》中说明了当我们要求没有无风险套利空间，选取  $p = \frac{e^{r\Delta t} - d}{u - d}$  时，期权在当前节点的价格为期权在分叉后价格期望的贴现后价格。此外我们需要选取的  $u$  和  $d$  会让股票价格的波动率与几何布朗运动：

$$\frac{dS}{S} = rdt + \sigma dz . \quad (1)$$

相符合。即考虑下列等式，

$$pu + (1-p)d = e^{r\Delta t}, \quad p(u-1)^2 + (1-p)(d-1)^2 - [p(u-1) + (1-p)(d-1)]^2 = \sigma^2\Delta t . \quad (2)$$

\*作者不对内容正确性负责。如果您希望使用部分内容作为报告、文章内容等，请您注明内容来源为“金融工程资料小站”网站。

让  $d = 1/u$ ，我们会得到一个重合的树形。这时忽略  $\Delta t$  的  $\frac{3}{2}$  阶小量后会有

$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = e^{-\sigma\sqrt{\Delta t}}. \quad (3)$$

上面左图和右图分别为一步和多步二叉树分叉过程，多步二叉树中每一个节点的分叉过程都是相同的。

## 2 二叉树计算期权价格步骤

欧式期权价格：

1. 根据给定的股票价格的波动率  $\sigma$  和无风险利率  $r$ ，按照上图中公式计算出  $u, d$  和  $p$ ，其中  $\Delta t$  为每一步步长  $T/N$ 。
2. 计算出股票价格在树形末端每个节点的价格，如上右图右侧。
3. 根据期权类型（看涨或看跌），以及期权的执行价格  $K$  计算出期权在树形末端的价格分布。
4. 用期权价格的递推关系计算出前一节点上期权价格的期望值。期权价格的递推关系为

$$f_{i,j} = e^{-r\Delta t}(f_{i+1,j+1}p + (1-p)f_{i+1,j}). \quad (4)$$

其中  $f_{i,j}$  为期权在  $i\Delta t$  时刻，第  $j$  层（由下往上数，从 0 开始）的价格。

5. 重复过程 4，直到计算出根节点处的期权价格  $f_{0,0}$ 。

美式期权的价格，重复欧式期权价格的计算过程 1 至 3，然后在递推上一层期权价格时考虑可以在该节点执行的情况，具体为：

1. 根据给定的股票价格的波动率  $\sigma$  和无风险利率  $r$ ，按照上图中公式计算出  $u, d$  和  $p$ 。
2. 计算出股票价格在树形末端每个节点的价格。
3. 根据期权类型（看涨或看跌），以及期权的执行价格  $K$  计算出期权在树形末端的价格分布。
4. 计算出前一层节点上期权价格的期望值。此时需要考虑可以在该节点执行期权，假设期权为看涨期权，则其价格的递推关系为

$$f_{i,j} = \max[S_{i,j} - K, e^{-r\Delta t}(f_{i+1,j+1}p + (1-p)f_{i+1,j})]. \quad (5)$$

其中  $f_{i,j}$  为期权在  $i\Delta t$  时刻，第  $j$  层的价格， $S_{i,j}$  为该处的股票价格。

5. 重复过程 4，计算出根节点处期权的价格  $f_{0,0}$ 。

## 3 步骤 Python 代码实现

---

```
import numpy as np 1
E = np.e 2
3
class Binomial_tree_sim: 4
    def __init__(self, r, sigma, S_0, K, T, steps, option_type="european", call_or_put="call"): 5
        """ 用输入参数初始化树。 6
        """ 7
        self.r = r 8
        self.sigma = sigma 9
        self.S_0 = S_0 10
        self.K = K 11
        self.T = T 12
```

```

self.steps = steps
self.option_type = option_type
self.call_or_put = call_or_put

# 计算出树形分叉参数。
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)

# 将会得到的结果。
self.tree = None
self.option_price = None

# 计算出一个树形。
self.build_tree()

def build_tree(self):
    """ 计算出股票价格在树形上每个节点的价格。 """
    self.tree = list()
    for lvl in range(self.steps+1):
        row = list()
        for j in range(lvl+1):
            node = dict()
            node["stock_price"] = self.S_0*self.u**(j)*self.d**(lvl-j)
            node["option_price"] = None
            row.append(node)
        self.tree.append(row)
    return

def calculate_option_price(self):
    """ 计算给定类型期权的价格。 """
    # 如果是欧式期权。
    if self.option_type == "european":
        # 计算出期权在树形末端的价格。
        for node in self.tree[-1]:
            # 如果是看涨期权。
            if self.call_or_put == "call":
                node["option_price"] = max(node["stock_price"]-self.K, 0)
            # 如果是看跌期权。
            else:
                node["option_price"] = max(self.K-node["stock_price"], 0)
        # 递推出树形根节点期权的价格。
        for lvl in range(self.steps-1, -1, -1):
            for j in range(len(self.tree[lvl])):
                self.tree[lvl][j]["option_price"] = E**(-self.r*self.dt)*(self.p*self.tree[lvl+1][j+1]\
                    ["option_price"]+(1-self.p)*self.tree[lvl+1][j]["option_price"])
    # 如果是美式期权，过程同欧式期权，计算节点价格时考虑需不需要在该节点执行。
    else:
        for node in self.tree[-1]:

```

```

node["option_price"] = max(node["stock_price"]-self.K, 0)
else:
node["option_price"] = max(self.K-node["stock_price"], 0)
for lvl in range(self.steps-1, -1, -1):
    for j in range(len(self.tree[lvl])):
        self.tree[lvl][j]["option_price"] = E**(-self.r*self.dt)*(self.p*self.tree[lvl+1][j+1]\
            ["option_price"]+(1-self.p)*self.tree[lvl+1][j]["option_price"])
        # 考虑需不需要在这时执行。
        if self.call_or_put == "call":
            self.tree[lvl][j]["option_price"] = max(self.tree[lvl][j]["option_price"], \
                self.tree[lvl][j]["stock_price"]-self.K)
        else:
            self.tree[lvl][j]["option_price"] = max(self.tree[lvl][j]["option_price"], \
                self.K-self.tree[lvl][j]["stock_price"])

self.option_price = self.tree[0][0]["option_price"]
return

```

## 4 计算示例

考虑一个期限在 3 年后，执行价格为 10 的期权。当前股票价格为 10，无风险利率为 0.05，股票价格波动率为 0.2。使用 10 步二叉树，可以如下计算出各种期权价格。

```

tree_obj = Binomial_tree_sim(0.05, 0.2, 10, 10, 3, 10, option_type="european", call_or_put="call")
tree_obj.calculate_option_price()
print("欧式看涨期权价格为: {:.4f}".format(tree_obj.option_price))

tree_obj = Binomial_tree_sim(0.05, 0.2, 10, 10, 3, 10, option_type="european", call_or_put="put")
tree_obj.calculate_option_price()
print("欧式看跌期权价格为: {:.4f}".format(tree_obj.option_price))

tree_obj = Binomial_tree_sim(0.05, 0.2, 10, 10, 3, 10, option_type="american", call_or_put="call")
tree_obj.calculate_option_price()
print("美式看涨期权价格为: {:.4f}".format(tree_obj.option_price))

tree_obj = Binomial_tree_sim(0.05, 0.2, 10, 10, 3, 10, option_type="american", call_or_put="put")
tree_obj.calculate_option_price()
print("美式看跌期权价格为: {:.4f}".format(tree_obj.option_price))

Output:
欧式看涨期权价格为: 2.0585
欧式看跌期权价格为: 0.6656
美式看涨期权价格为: 2.0585
美式看跌期权价格为: 0.8563

```

## 5 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 21 章，John C. Hull 著，王勇、索吾林译，机械工业出版社，2014.11。