8 蒙卡模拟计算美式期权价格 (a)

使用申明*

2021 年 2 月 23 日

目录

1 简介

1.1 蒙卡模拟美式期权

美式期权的持有者可以在期权有效期内任一时刻选择执行期权。如果直接通过蒙特卡罗模拟方法模拟股价变化路径，由于在路径中并不知道当前股价对应的美式期权价格，所以无法判断应不应该执行期权，也因此无法直接使用蒙卡模拟方法估计美式期权价格。

不过我们可以在对股价变化过程离散化的同时，把可以执行美式期权的时间固定在这些离散的时间点。然后首先确定期权在每条路径末端执行时刻的价格，再一步步向前递推出上一时刻的期权价格，直到得到初始时刻期权价格，每条路径初始时刻期权价格的平均值即为模拟计算出的结果。期权价格的递推通过贴现，最小二乘法线性拟合，判断是否执行，三步实现。

由于普通美式看涨期权在执行时刻之前总是不会被提前执行，所以下面考虑的期权为美式看跌期权。

1.2 美式看涨期权不会被提前执行

假设我们当前持有一个实值（股价大于执行价格）美式看涨期权。如果我们执行期权，即以价格 K 买入相应股票，我们下一步只可能：

A 继续持有股票到某一时刻 t_1，那么这种情况下，就不如不执行期权，因为不执行期权的话可以先将 K 的资金做无风险投资，然后在相同的时刻 t_1 再以 K 的价格买入股票，这样会有额外 K 资金的利息收入。

B 直接卖出该股票，则我们立刻会有一个 $S-K$ 的现金收入。但是这么做不如直接将该美式看涨期权卖出。由于美式看涨期权具有时间价值，所以其价格总是高于 $\max(S-K,0)$，或者由 $C \geq c \geq S - Ke^{-rT} > S - K$ 看出。

因此，总是不应该提前执行美式看涨期权。

2 最小二乘法计算美式期权价格步骤

1. 根据给定参数确定股价离散化变化过程：

$$S(t+\Delta t) = S(t)\, e^{(r-\frac{\sigma^2}{2})\Delta t + \sigma\varepsilon\sqrt{\Delta t}}, \quad \varepsilon \sim N(0,1), \quad \Delta t = T/M\,. \tag{1}$$

2. 按照股价离散化变化过程抽样出 N 条股价变化路径保存下来。

3. 确定每条路径末端期权价格。

4. 将每条路径上期权价格贴现到该路径上上一个时间节点。在该时间节点，每条路径上的股价都会对应一个贴现后的参考期权价格。

5. 对该时间节点上所有股价和参考期权价格进行线性回归拟合。记参考期权价格为 V_i，线性拟合出的期权价格为 V_i^*，$V_i^* = a + bS_i + cS_i^2$。求出使 $\sum_{i=1}^{N}(V_i - V_i^*)^2 = \sum_{i=1}^{N}(V_i - a - bS_i - cS_i^2)^2$ 取极小值的 a, b, c，然后用该表达式计算出该时刻每个股价对应期权价格 V_i^*，以此取代由下一个时刻期权价格贴现来的参考期权价格。

6. 最后判断在该时刻是不是应该执行期权，如果执行期权的收益大于拟合表达式计算出的期权价格，则该处期权价格更新为执行期权后的收益。

7. 重复步骤 4，5，6，直到计算出初始时刻之后一时刻每条路径样本上的期权价格。取算术平均值并贴现到初始时刻后，即为蒙卡模拟计算出的美式期权价格。

3 步骤 Python 代码实现

```
import numpy as np

def sample_paths(r, sigma, S_0, T, M, N):
    data = S_0*np.ones((N,1))
    for i in range(M):
        normal_variables = np.random.normal(0, 1, (N, 1))
        ratios = np.exp((r-0.5*sigma*sigma)*T/M+normal_variables*sigma*(T/M)**0.5)
        # data[:, -1:] 截取的数据维度形状可以保持为(N, 1), 而data[:, -1]截取的数据的维度为(N)。
        data = np.concatenate((data, data[:, -1:]*ratios), axis=1)

    return data

def linear_fitting(X, Y):
    X = np.array(X)
    Y = np.array(Y)
    S0, S1, S2, S3, S4 = len(X), sum(X), sum(X*X), sum(X**3), sum(X**4)
    V0, V1, V2 = sum(Y), sum(Y*X), sum(Y*X*X)
    coeff_mat = np.array([[S0, S1, S2], [S1, S2, S3], [S2, S3, S4]])
    target_vec = np.array([V0, V1, V2])

    inv_coeff_mat = np.linalg.inv(coeff_mat)
    fitted_coeff = np.matmul(inv_coeff_mat, target_vec)
```

```
    resulted_Ys = fitted_coeff[0]+fitted_coeff[1]*X+fitted_coeff[2]*X*X

    return resulted_Ys

def MC_American_put_price(r, sigma, S_0, K, T, M, N):
    # data数组的维度形状为(N, M+1), 即(paths_num, steps_num+1)。
    data = sample_paths(r, sigma, S_0, T, M, N)
    option_prices = np.maximum(K-data[:, -1], 0)

    for i in range(M-1, 0, -1):
        # 期权价格贴现到当前时刻。
        option_prices *= np.exp(-r*T/M)
        # 线性回归拟合更新期权价格。
        option_prices = linear_fitting(data[:, i], option_prices)
        #判断是不是应该执行期权，并更新价格。
        option_prices = np.maximum(option_prices, K-data[:, i])
    # 递推回的是初始时刻下一时刻，所以还需要再贴现一次。
    option_prices *= np.exp(-r*T/M)

    return np.average(option_prices)
```

4 计算示例

考虑无风险利率为 0.1，股价波动率为 0.4，初始股价为 50，执行时间为 5 个月后，执行价格为 60 的美式看跌期权。我们把股价变化过程离散化为 $M = 200$ 小段，进行蒙卡抽样股价变化路径 $N = 40000$ 条。计算出的美式看跌期权价格为：

```
r, sigma, S_0, K, T = 0.1, 0.4, 50, 60, 5/12
M, N = 200, 40000
Ame_put_price = MC_American_put_price(r, sigma, S_0, K, T, M, N)
print("蒙卡模拟美式看跌期权价格为：{:.5f}".format(Ame_put_price))

Output:
    蒙卡模拟美式看跌期权价格为：11.11454
```

如果使用同样的参数，使用树形计算该美式看跌期权价格，用 400 步树形，会得到结果为 10.85362。由于使用树形计算普通美式期权的精确度比较高，所以此处蒙卡模拟计算的结果 11.11454 是偏高的。

5 相关说明

5.1 最小二乘法线性回归拟合期权价格

上面计算中，我们将下一时刻的期权价格贴现到当前时刻后，使用了最小二乘法对每条路径上的股票价格和期权参考价格进行线性拟合。我们具体使用的表达式为：

$$V_i^* = a + bS_i + cS_i^2 \ . \tag{2}$$

其中参数的确定，是考虑最小化平方误差，

$$a, b, c = \mathrm{Arg}\min_{a,b,c} \sum_{i=1}^{N} (V_i - a - bS_i - cS_i^2)^2 = \mathrm{Arg}\min_{a,b,c} \mathcal{L} \ . \tag{3}$$

$$\frac{\partial \mathcal{L}}{\partial a} = 0, \quad \frac{\partial \mathcal{L}}{\partial b} = 0, \quad \frac{\partial \mathcal{L}}{\partial c} = 0 \ . \tag{4}$$

即我们会需要解方程组，

$$\begin{pmatrix} N & \sum_i S_i & \sum_i S_i^2 \\ \sum_i S_i & \sum_i S_i^2 & \sum_i S_i^3 \\ \sum_i S_i^2 & \sum_i S_i^3 & \sum_i S_i^4 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_i V_i \\ \sum_i V_i S_i \\ \sum_i V_i S_i^2 \end{pmatrix} . \tag{5}$$

记为

$$C \cdot A = V . \tag{6}$$

则系数 $A = (a, b, c)^T = C^{-1}V$。我们可以使用 numpy.linalg.inv() 对矩阵求逆，按上式得到最小二乘法线性拟合系数。然后在一固定时刻，每条路径上的股票价格都可以计算出一个线性回归期权价格。

但是这里的二阶线性方程最小二乘法拟合存在两个问题：

1. 当股票价格比较大时，比如股票价格在 100 附近，则矩阵中的 N 和 $\sum_i S_i^4$ 元素将差 10^8 的量级，而且其它元素间量级差距也比较大，事实上数值求得的逆矩阵 C^{-1} 的精确度可能会不高。
2. 当初始时刻美式看跌期权不是深度实值时，用一个二阶多项式拟合的结果会给出偏高的期权价格估计。这是由于二阶多项式曲线形状的局限。

其中第二个问题我们可以由下图说明。该图为期权有效期中间某一时刻，我们对每条路径上股价和期权价格进行线

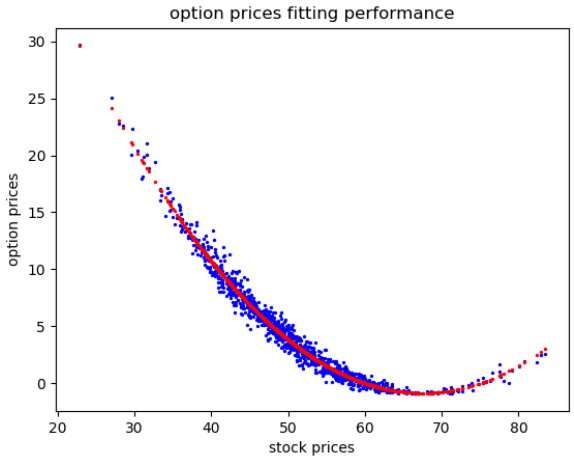

Fig. 1

性回归拟合的结果。其中蓝色点为下一时刻期权价格贴现来的参考期权价格，红色点为二阶多项式最小二乘法线性拟合结果。可以看到当股价大于一定值之后，期权价格反而有上升，这是不合理的。因为当股价增加，美式看跌期权的价格总是应该变得更低。这也导致了模拟计算出的美式看跌期权的价格偏高。

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 。