5 Calculate Asian Option Price by Binomial Tree Method

declaration*

05/01/2021

Contents

1 Introduction

1.1 Asian Option

The price of asian option is determined by the changing path of stock prices before expiration date. Specifically, for asian average price call option, the option price at exercise time is $\max(S_{ave} - K,\ 0)$, where K is the agreed strike price, S_{ave} is the average of historical stock price. For average price put option, its price at exercise time is $\max(K - S_{ave},\ 0)$.

1.2 Classification of Asian Option

1. Average price option: as illustrated in the above, the prices at exercise time of average call and put options are $\max(S_{ave} - K,\ 0)$ and $\max(K - S_{ave},\ 0)$ respectively.
2. Average strike option: for average strike option, the price of call and put options at exercise time are $\max(S_T - S_{ave},\ 0)$ and $\max(S_{ave} - S_T,\ 0)$ respectively, where S_T is the stock price at exercise time T.
3. American/European option: either average price or average strike option, can be American type or European type. If the option is American type, the option holder has the right to exercise the option at any time before expiration date.

2 Calculation Procedures of Asian Option Price by Tree Method

First as always we construct a binomial tree to describe the stock price changing process. Then for each node, calculate the min and max values of possible stock prices paths arriving at this node, interpolate a number of value points between min and max values with equal space. Then according to the specific type of the asian option, on leaf level calculate option prices corresponds to the value points between each node's min and max historical average stock prices. And calculate backwardly to the root node. While doing backward calculation, interpolation method is needed to calculate every average stock price's option price. Here we use a number of value points instead of all possible historical paths' average stock prices, is because the number of all possible average stock prices may be very large.

Detailed procedures are:

1. From given parameters calculate the p, u and d of the binomial tree, and build the stock price changing binomial tree.
$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = \frac{1}{u}, \quad p = \frac{e^{r\Delta t} - d}{u - d}\ . \tag{1}$$
2. Calculate and save the historical paths' max average stock price $S_{ave_max}$and min average stock price S_{ave_min} at each node on the tree,
$$S_{ave\_max} = \frac{1}{i+1}\left[S_0\frac{1-u^{j+1}}{1-u} + S_0u^jd\,\frac{1-d^{i-j}}{1-d}\right],$$
$$S_{ave\_min} = \frac{1}{i+1}\left[S_0\frac{1-d^{i-j+1}}{1-d} + S_0d^{i-j}u\,\frac{1-u^{j}}{1-u}\right].$$

*The author is not responsible for the correctness of the contents. If you would like to use part of the contents in your presentation, article, etc., please add the website "Quantitative Finance Numerics" as a reference.

Where S_0 is the initial stock price, i is the level index of the node (0 for root node), j is the position index on this level (0 for min stock price). Then interpolate a fixed number of value points between min and max historical average prices.

3. On the leaf level of the tree, according to the specific type of the option, calculate the option prices to all the historical average price value points, and save the results.

4. Do backwards induction to previous level. Calculate the option prices correspond to current level's each node's every historical average stock price value points. Consider a specific average stock price value point, after stock price's increase or decrease in next step, it will arrive at an average stock price value point at next level. However this arrived point may not be inside the calculated value points, in general we need to use interpolation method to calculate the option price corresponds to the arrived value point. The weighted and discounted option prices correspond to the value points after stock price's increasing or decreasing, is the current level's node's average stock price value point's option price.

5. Repeat procedure 4, until we get the option price at the root node.

If the option is American type, then in procedure 4, we need to consider whether to exercise the option or not at each node's each historical average stock price value point. The other part are the same.

3 Implementation with Python

```
import math

E = math.e

class Tree_asian_option:
    def __init__(self, r, sigma, S_0, K, T, steps, points):
        """ Initialize the instance, points is number of value points of each node's historical average stock
            price.
        """
        self.r = r
        self.sigma = sigma
        self.S_0 = S_0
        self.K = K
        self.T = T
        self.steps = steps
        self.points = points

        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)

        self.call_price = None
        self.tree = list()
        self.build_tree()

    def get_max_min_ave(self, i, j):
        """ Calculate the max and min historical average stock prices at node (i, j).
        """
        u, d, S_0 = self.u, self.d, self.S_0
        max_ave = S_0*(1-u**(j+1))/(1-u)+S_0*(u**j)*d*(1-d**(i-j))/(1-d)
        max_ave /= 1+i
        min_ave = S_0*(1-d**(i-j+1))/(1-d)+S_0*(d**(i-j))*u*(1-u**j)/(1-u)
        min_ave /= 1+i

        return (max_ave, min_ave)

    def interpolation(self, x, ref_list):
        """ Linear interpolation, ref_list is a list with dimension points x 2,
            ref_list is default to be sorted by ref_list[i][0] with increasing order.
        """
```

```
        left , right = 0, len( ref_list )-1
        pos = 0

        # If value is out of the range, return the boundary value.
        if x > ref_list [ right ][0]:
            return ref_list [ right ][1]
        if x < ref_list [ left ][0]:
            return ref_list [ left ][1]

        # Find x's position at {ref_list [ i ][0]} by binary search.
        while left < right:
            pos = int(( left +right)/2)
            if x == ref_list[pos ][0]:
                return ref_list [pos ][1]
            if x > ref_list [pos ][0]:
                left = pos+1
            else:
                right = pos-1
        if x > ref_list [ left ][0]:
            pos = left+1
        else:
            pos = left

        # Linear interpolation.
        result = (ref_list [pos ][1]- ref_list [pos -1][1]) /( ref_list [pos ][0]- ref_list [pos -1][0])
        result *= (x- ref_list [pos -1][0])
        result += ref_list[pos -1][1]

        return result

    def build_tree(self):
        S_0, steps, points = self.S_0, self.steps, self.points
        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["F_S"] = list()
                max_ave, min_ave = self.get_max_min_ave(lvl, j)
                for k in range(points):
                    node["F_S"].append([(max_ave-min_ave)/(points-1)*k+min_ave, None])
                row.append(node)
            self.tree.append(row)

        return

    def calculate_call_price(self):
        """ Calculate European average price call option's price.
        """
        r, S_0, K = self.r, self.S_0, self.K
        steps, points = self.steps, self.points
        dt, u, d, p = self.dt, self.u, self.d, self.p

        a = E**(-r*dt)
        # Boundary condition.
        for node in self.tree [-1]:
            for k in range(points):
                node["F_S"][k][1] = max(0, node["F_S"][k][0]-K)
```

```
        # Backwards induction to the root node.
        for lvl in range(steps-1, -1, -1):
            for j in range(lvl+1):
                node = self.tree[lvl][j]
                for k in range(points):
                    new_ave_u = (node["F_S"][k][0]*(lvl+1)+self.tree[lvl+1][j+1]["S"])/(lvl+2)
                    new_ave_d = (node["F_S"][k][0]*(lvl+1)+self.tree[lvl+1][j]["S"])/(lvl+2)
                    option_price_u = self.interpolation(new_ave_u, self.tree[lvl+1][j+1]["F_S"])
                    option_price_d = self.interpolation(new_ave_d, self.tree[lvl+1][j]["F_S"])
                    node["F_S"][k][1] = a*p*option_price_u+a*(1-p)*option_price_d

        self.call_price = self.tree[0][0]["F_S"][0][1]

        return
```

4 Calculation Example

Suppose the risk-free rate is 0.1, volatility of stock price is 0.4, initial stock price is 50. Consider an average price call option, exercise time is one year later, exercise price is 50. We use 60 steps binomial tree, each node's historical average prices are represented by 100 equally spacing value points. Calculation is the following:

```
print("r=0.1, sigma=0.4, S_0=50, K=50, T=1, steps=60, points=100 . \n")
tree_obj = Tree_asian_option(0.1, 0.4, 50, 50, 1, 60, 100)
tree_obj.calculate_call_price()
print("European average price call option price: {0:.5f}".format(tree_obj.call_price))

Output:
    r=0.1, sigma=0.4, S_0=50, K=50, T=1, steps=60, points=100 .
    European average price call option price: 5.57973
```

5 Relavent Clarification

5.1 the Max and Min Values of Historical Average Prices

For the level i index j node (i, j), ignore the order of decrease or increase, upon arriving at this node, stock price has went through j times increase, $(i - j)$ times decrease. For the paths arriving at this node, the case that the average price takes the max value, corresponds to a path that the stock price first increase j times then decrease $(i - j)$ times. The case that the average price takes the min value, corresponds to a path that the stock price first decrease $(i - j)$ times then increase j times. As the following figures:

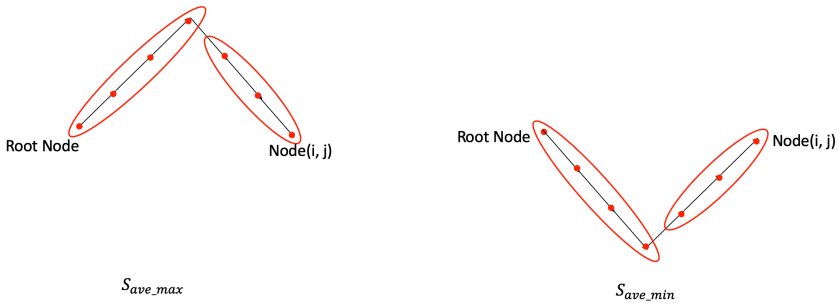

Fig. 1

$$S_{ave\_max} = \frac{1}{i+1}\left[S_0\frac{1-u^{j+1}}{1-u} + S_0u^jd\frac{1-d^{i-j}}{1-d}\right], \tag{2}$$

$$S_{ave\_min} = \frac{1}{i+1}\left[S_0\frac{1-d^{i-j+1}}{1-d} + S_0d^{i-j}u\frac{1-u^j}{1-u}\right]. \tag{3}$$

5.2 Backwards Induction of Option Prices

Node (i, j) is self.tree[i][j] in codes, it's a dict() type variable. It has elements "S" and "F_S", the value in "S" is current stock price $S_{i,j}$, "F_S" contains a two dimensional list() variable, which stores $N = points$ pairs of average stock price value points and corresponded option price. The list of "F_S" discretely describes a function from historical average stock prices to option prices.

Consider an average stock price value point $S_{ave}(k) = \frac{k}{N-1}(S_{ave\_max} - S_{ave\_min}) + S_{ave\_min}$,we want to calculate the corresponded option price. Since the stock price at current node has probability p to increase, probability $(1-p)$ to decrease. If the stock price increases, the historical average price will become $\frac{1}{i+2}(S_{ave}(k) \times (i+1) + uS_{i,j})$, we can use the "F_S" list in next level's $(i+1, j+1)$node and apply interpolation to get the option price. If the stock price decreases, the historical average price will become $\frac{1}{i+2}(S_{ave}(k) \times (i+1) + dS_{i,j})$, we can use the "F_S" list in next level's $(i+1, j)$ node and apply interpolation to get the option price. The option corresponds to current node's $S_{ave}(k)$ is the weighted and discounted result of the two calculated option prices in the above.

6 Reference

References

[1] John C. Hull. chapter 27 of *Options, Futures and Other Derivatives*, 9th Edition, Pearson Education.