3 Calculate Lookback Option Prices by Binomial Tree Method

declaration*

04/20/2021

Contents

1 Introduction

1.1 Lookback Option

Lookback option is a type of option that the price depends on the historical min/max stock price. If the lookback option is European type option that can only be exercised at maturity date, then the option price has explicit solution. But here we mainly use binomial tree method numerically calculate current lookback option prices.

For floating lookback option, the price at maturity date is $\max(S_T - S_{min},\ 0)$, where S_T is the stock price at exercise date T, S_{min} is the historical stock min price before exercise date. Similarly, the price of floating lookback put option at maturity is $\max(S_{max} - S_T,\ 0)$, where S_{max} is the historical max stock price before maturity date.

1.2 Classification of Lookback Option

1. floating/fixed lookback option: the floating lookback option's payoff is described in the above paragraph. Fixed lookback option is similar to vanilla option, for fixed lookback call option, the payoff is the value the max stock price before maturity exceeds the agreed strike price. For fixed lookback put option, the payoff is the value that the agreed strike price exceeds the min stock price before maturity.

2. European/American lookback option: either floating or fixed lookback option, can be European or American type. If the option is American type option, the option holder has the right to exercise the option at any time before maturity date.

2 Numerical Calculation Procedures

We take European floating lookback call option as an example, illustrate the calculation procedures, First, we build a binomial tree to describe the stock price changing process. For vanilla European option binomial tree pricing, we know that at each node on the tree, the stock price at the node corresponds to an option price. However, for floating lookback call option, each stock price does not correspond to an option price. But each possible historical min stock price at each node corresponds to an option price, so for each node, we need to calculate and save all possible historical min stock prices. Then at the end level of the tree, at the maturity time, calculate the option

*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.

price for each node's each possible historical min stock price, and get the lookback option price at root node by backwards induction. In the backwards induction, we use every node's every historical min stock price, consider possible arrivable historical min stock prices' corresponding option prices in next level's nodes, being weighted and discounted.

Detailed procedures are:

1. From given parameters, calculate the binomial tree's branching parameters p, u and d. Build a binomial tree which describes stock prices' changing process.

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

2. From root node, calculate level by level, at each node on each level, calculate all possible historical min stock prices and save the values.

3. At the leaf level, calculate the corresponding option price for each node's each possible historical min stock prices and save the values.

4. Do the induction backwardly, calculate the corresponding option prices for each node's each possible historical min stock prices on current level. Specifically at each node, every possible historical min stock price corresponds to a class of paths, in this class of paths, whether stock price rise or fall in next step, will correspond to some next level's nodes' historical min stock prices. So for current node's historical min stock price's corresponding option price, we can get it by weighting and discounting the arrived next level's nodes' historical min stock prices' option prices after increasing or decreasing current stock price.

5. Repeat procedure 4, until option price at root node is calculated.

For American type lookback call option, in procedure 4, we need to consider whether we should exercise the option under every node's every historical min stock price, other parts of procedures are the same as above.

3 Procedures Implementation with Python

```
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"):
        """ Initialize the instance.
        """
        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)
        # If strie_price is None, the option is floating lookback option,
        # otherwise it's fixed lookback option (strike price is strike_price).
        self.strike_price = strike_price
        self.call_price = None
        self.put_price = None
        # Build a tree.
        self.tree = list()
        self.build_tree()

    def build_tree(self):
        """ Build stock price changing tree, calculate every node's all possible historical min/max
            stock prices.
        """
```

```
        # Shorten parameters' names.
        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()
                # Find out every node's possible historical min and max values,
                # and initialize corresponding option price to be 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):
        """ Calculate lookback call option price.
        """
        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
        # Discount factor for each step.
        a = E**(-r*dt)
        # Calculate option prices at the end level of the tree.
        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)
        # Calculate option price back to the root node by backwards induction.
        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):
        """ Calculate lookback put option price.
        """
        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 Calculation Example

Suppose risk-free rate is 0.1, volatility is 0.4, stock's initial price is 50, exercise time is 3 months later, consider several types of lookback option. Calculate option prices by 5-step binomial tree. For fixed lookback option, we choose strike price to be 49. Calculation results are the following.

```
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("European floating call: {0:.5f}".format(tree_obj.call_price))
print("European floating put: {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("American floating call: {0:.5f}".format(tree_obj.call_price))
```

```
print("American floating put: {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("European fixed call: {0:.5f}".format(tree_obj.call_price))
print("European fixed put: {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("American fixed call: {0:.5f}".format(tree_obj.call_price))
print("American fixed put: {0:.5f}".format(tree_obj.put_price))

Output:
r=0.1, sigma=0.4, S_0=50, T=0.25, steps=5.
European floating call: 6.48347
European floating put: 5.69116
American floating call: 6.48347
American floating put: 5.91857

r=0.1, sigma=0.4, S_0=50, T=0.25, steps=5, strike_price=49 .
European fixed call: 7.90097
European fixed put: 4.58603
American fixed call: 7.92152
American fixed put: 4.59751
```

5 Relavent Clarification

5.1 Calculation of Node's Historical min/max Stock Prices

Consider any node, on the binomial tree which describes the stock price's changing, there are possible paths from root node to current node, each possible path has a max stock price and a min stock price. A node's all possible historical min/max stock prices are, the historical min/max stock prices on all possible paths arriving at this node. Suppose level index i counts from 0, node index j counts from 0 (bottom up). For the nodes on level i, the stock price at each node has increased and decreased i times in total. Increasing amplitude u and decreasing amplitude d satisfy equation $ud = 1$.

Suppose we want to find out all possible historical min stock prices of the node at level i index j. Ignore the ordering, the node at level i index j has went through j times increase, $(i-j)$ times decrease. The minimum value of historical min prices to this node is $S_0 d^{i-j}$, corresponds to a path first decrease $(i-j)$ times, then increase j times. And the maximum value of historical min prices to this node is $\min(S_0,\ S_0 u^j d^{i-j}) = \min(S_0,\ S_0 d^{i-2j})$, is the minimum of the stock prices at start and end points of the paths. So the paths' historical min values have range $[S_0 d^{i-j},\ \min(S_0,\ S_0 d^{i-2j})]$.Since the stock price can only have discrete values $S_0 d^k$, k is an integer, and for every value in $S_0 d^{i-j}, S_0 d^{i-j-1}, ..., \min(S_0,\ S_0 d^{i-2j})$, we can find a path connecting root node to current node in the binomial tree, which has historical min stock price to be this value. Thus for the node at level i index j, all possible historical min stock prices are $S_0 d^{i-j}, S_0 d^{i-j-1}, ..., \min(S_0,\ S_0 d^{i-2j})$.

Similarly, suppose we want to find out all possible historical max stock price of the node at level i index j. Ignore the ordering, the node has went through j times increase, $(i-j)$ times decrease. The minimum value of historical max prices of paths to this node is $\max(S_0,\ S_0 d^{i-2j})$, which is the maximum of start and end points stock prices. For node at level i index j, all possible historical max stock prices are $\max(S_0,\ S_0 d^{i-2j}), ..., S_0 d^{1-j}, S_0 d^{-j}$.

From the above discussion, we notice that for any node on the tree, all min/max values of historical stock prices of all paths to this node can be represented by $S_0 d^k$, k is an integer in some range. So in the codes implementation, we can use dictionary to store min/max stock price and corresponding option price, key is the k in the min/max stock price $S_0 d^k$, value is the related option price.

5.2 Backwards Induction of Option Prices

We take American floating lookback call option as an example. The option's price at maturity is the value that spot stock price exceeds historical min stock price. After build the stock price changing binomial tree and get all

possible historical min/max stock prices at each node. We can first calculate option prices at maturity date, i.e. calculate the option price at the leaf level on the tree. Then apply backwards induction towards root node level by level.

For floating lookback call option, the historical min/max price should be historical minimum stock price. Suppose we are on level i index j node, and are calculating the option price corresponds to historical min stock price $S_0 d^k$. The option price here is the weighted and discounted result of the option prices in possible status after on step branching. There's probability p, after on step branching we arrive at level $i+1$ index $j+1$ node, since stock price has increased, the historical min price is the same as previous node's historical min stock price, which is $S_0 d^k$. There's probability $(1-p)$, after one step branching, we arrive at level $i+1$ index j node, the historical min stock price become $\min(S_0 d^k,\ S_+ 0 d^{i+1-2j})$, because the decrease in this steps might make the min stock price become lower. Take the weighted average of the option prices in these two cases, and multiply the one step discount factor, we'll get the option price for the node's historical min stock price. If the option is American type, we need to compare the result with payoff of exercising the option under the historical min stock price, the final option price is the higher one. Using this method calculate every level's every node's every historical min stock price's option price, until arrive at root node.

For other types of lookback options, the procedures are similar, whether we should consider min or max historical stock prices, will depend on the specific option types.

6 Reference

References

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