Calculate European and American Option Prices by Binomial Tree Method

declaration*

03/05/2021

Contents

1 Introduction

Suppose the option is stock option, binomial tree is the graph that describes the possible stock prices changing paths before expiration. Let a binomial tree have time range 0 to T, branching steps N, and at time stamps $t = 0, .., (N-1)\Delta t$ stock price has probability p to change from current price S_t to uS_t and probability $(1-p)$ to change to dS_t. Where $\Delta t = \frac{T}{N}$, u and d are increasing and decreasing amplitudes respectively. When the branching steps of the binomial tree is sufficiently large, the final distribution of stock price will be log-normal distribution.

In chapter 13 of *Options, Futures and Other Derivatives*, it is proved that when we suppose no arbitrage, and let $p = \frac{e^{r\Delta t}-d}{u-d}$, the option price at current node will be the possible option prices after branching and being weighted and discounted. Besides, we should choose the u and d such that the volatility of stock price will match the geometric Brownian motion:

$$\frac{dS}{S} = rdt + \sigma dz \ . \tag{1}$$

I.e. the following equations should be satisfied,

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

Let $d = 1/u$, we will get a recombined tree. Then ignore terms smaller than or equal to $\Delta t^{\frac{3}{2}}$, and get

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

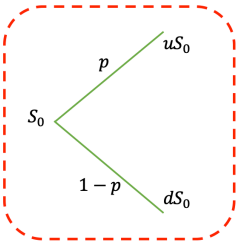
The following left figure and right figure are one-step and multiple-step binomial tree branching processes respectively. Every node's branching processes are the same in the multiple-step binomial tree.

2 Procedures for Option Price Calculation by Binomial Tree Method

European option price:

1. From the given stock price volatility σ and risk-free rate r, calculate u, d and p with the formulas in the above graph, where Δt is the step size T/N.
2. Calculate stock prices at each node on the last level of the tree, like the right side of the above right figure.
3. According to the option type (call or put), and option's strike price K, calculate the prices distribution of the option on the last level of the tree.

*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.



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

Fig. 1

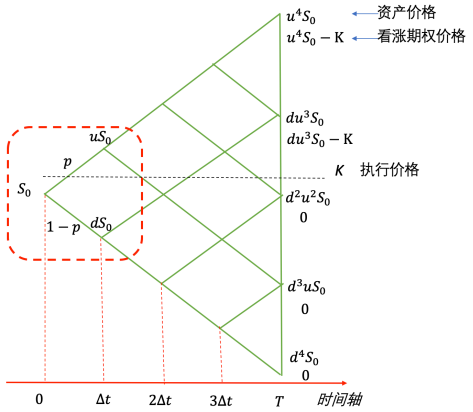


Fig. 2

4. Through backwards induction of the option prices, calculate the expected values of option prices on the previous level's nodes. The backwards induction of option prices is:

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

   Where $f_{i,j}$ is the price of the option at time $i\Delta t$, and index j (bottom up, from 0).

5. Repeat procedure 4, until the option price $f_{0,0}$ at the root node is calculated.

American option price, repeat European option price calculation procedures 1 to 3, then consider whether the option should be exercised when calculating previous level's prices backwardly, which are:

1. From the given stock price volatility σ and risk-free rate r, calculate u, d and p with formulas in the above graph.

2. Calculate stock prices at each node on the last level of the tree.

3. According to the option type (call or put), and option's strike price K, calculate the prices distribution of the option on the last level of the tree.

4. Calculate the expected values of option prices on previous level's nodes. Here it should be considered that the option might be exercised at current node, suppose the option is call option, then the price's backwards induction is:

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

   Where $f_{i,j}$ is the price of the option at time $i\Delta t$, index j, $S_{i,j}$ is the stock price at the same location.

5. Repeat procedure 4, until the option price $f_{0,0}$ at the root node is calculated.

3 Procedures Implementation with Python

```
import numpy as np
E = np.e

class Binomial_tree_sim:
    def __init__(self, r, sigma, S_0, K, T, steps, option_type="european", call_or_put="call"):
        """ Initialize the tree with input parameters.
        """
        self.r = r
        self.sigma = sigma
        self.S_0 = S_0
        self.K = K
        self.T = T
        self.steps = steps
        self.option_type = option_type
        self.call_or_put = call_or_put

        # Calculate tree branching factors.
```

```
        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)

        # Expected output.
        self.tree = None
        self.option_price = None

        # Calculate a tree.
        self.build_tree()

    def build_tree(self):
        """ Calculate stock prices at each node in the tree.
        """
        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):
        """ Calculate the option price with given type.
        """
        # If the option is European option.
        if self.option_type == "european":
            # Calculate the option prices at the last level of the tree.
            for node in self.tree[-1]:
                # If the option is call option.
                if self.call_or_put == "call":
                    node["option_price"] = max(node["stock_price"]-self.K, 0)
                # if the option is put option.
                else:
                    node["option_price"] = max(self.K-node["stock_price"], 0)
            # Calculate out the option price at root node by backwards induction.
            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"])
        # If the option is American option, similar procedures with European option,
        # but need to consider whether to exercise or not when calculating price at each nodes.
        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"])
                    # Consider if we should exercise or not.
                    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 Calculate Example

Consider an option which will be exercised 3 years later, with strike price 10. Current stock price is 10, risk-free rate is 0.05, stock price volatility is 0.2 . Using 10-step binomial tree, different types of option prices can be calculated as following.

```
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("European call option price: {:0.4f}".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("European put option price: {:0.4f}".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("American call option price: {:0.4f}".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("American put option price: {:0.4f}".format(tree_obj.option_price))

Output:
European call option price: 2.0585
European put option price: 0.6656
American call option price: 2.0585
American put option price: 0.8563
```

5 Reference

References

[1] John C. Hull. chapter 13, 21 of *Options, Futures and Other Derivatives*, 9th Edition, Pearson Education.