1 European Option Price

declaration*

03/01/2021

Contents

1 Introduction

Suppose the asset price is $S(t)$, or S, its changing process is a geometric Brownian motion,

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

The drift rate μ and volatility σ are set to be constants, dz is a Wiener process (Brownian motion) random term.

We observe at time $t = 0$, get the asset price S_0. And suppose the European call or put option we are considering has exercise time T and strike price K. By the Black-Scholes-Merton European option pricing formula, we know the call option price c and put option price p can be represented as:

$$c = S_0 N(d_1) - K e^{-rT} N(d_2), \quad p = K e^{-rT} N(-d_2) - S_0 N(-d_1) \ . \tag{2}$$

Here, r is the risk-free rate, which is set to be constant, function $N(x)$ is the normal distribution cumulative probability function, and d_1 and d_2 are

$$d_1 = \frac{\ln \frac{S_0}{K} + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = \frac{\ln \frac{S_0}{K} + (r - \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T} \ . \tag{3}$$

For numerical calculation with Python, we only need to use two modules "numpy" and "scipy". The detailed implementation is fairly easy, just rewrite the pricing formula into Python function.

2 Implementation with Python

```
import numpy as np
from scipy.stats import norm

E = np.e

# A set of parameters as an example.
r, sigma, S_0, K, T = 0.05, 0.20, 90.0, 100.0, 3.0

# European option price function.
def european_option(r, sigma, S_0, K, T):
    """ r: risk-free rate; sigma: volatility of asset price;        S_0: initial asset price;
```

*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.

```
    K: option strike price;        T: option exercise time.
"""
d_1 = (np.log(S_0/K)+(r+0.5*sigma*sigma)*T)/sigma/T**0.5
d_2 = d_1-sigma*T**0.5
call_price = S_0*norm.cdf(d_1)-K*E**(-rT)*norm.cdf(d_2)
put_price = K*E**(-r*T)*norm.cdf(-d_2)-S_0*norm.cdf(-d_1)

return (call_price, put_price)
```

Then we can verify whether the put-call parity equation is satisfied:

$$c + Ke^{-rT} = p + S_0 \ . \tag{4}$$

Suppose $r = 0.05,\ \sigma = 0.20,\ S_0 = 90,\ K = 100,\ T = 3$, apply the above function we can calculate and get:

```
call_price, put_price = european_option(r, sigma, S_0, K, T)
print("European call option price: {0:.5}, European put option price: {1:.5}".format(call_price, put_price))
print("Put-call parity: ")
print("    {0:.5}+{1:.5}={2:.5}+{3:.5}".format(call_price, K*E**(-r*T), put_price, S_0))
print("              {0:.5}={1:.5}".format(call_price+K*E**(-r*T), put_price+S_0))

Output:
    European call option price: 14.17, European put option price: 10.24 .
    Put-call parity:
        14.17+86.071=10.24+90.0
              100.24=100.24
```

3 Related Clarification

3.1 Price and Value

It feels like these two words are commonly used interchangeably and easily to be confused with. Therefore we give a bit clarification about their features:

(a) Price: usually, it's the bid price or trading price of the asset we are considering in the market.

(b) Value: the "reasonable" price of the asset, it's not strictly equal to the asset price in the market. "Reasonable" here means we can use reliable pricing methods (e.g. no arbitrage), through observing variables' values in the market other than the considered asset price, to estimate the asset's deserved price at some specific time.

Though it is fine to use them interchangeably in many cases, they should be different.

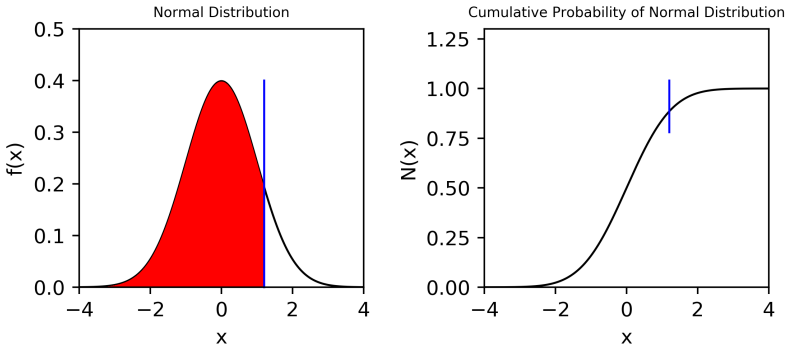

Fig 1

3.2 Normal Distribution Cumulative Probability Function

In the above option price expression, the function $N(x)$ is the normal distribution cumulative probability function, which is the integration of standard normal distribution (mean 0, std 1) density function from $-\infty$ to x, i.e.

$$N(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{s^2}{2}} \, ds \, . \tag{5}$$

Usually the function value is obtained by checking value table or numerical integration. Here we can just use the "stats.norm" class in the "scipy" module, this class contains some normal distribution related functions. In which, the "norm.cdf()" is the cumulative probability function. In the above figures, the left figure shows the normal distribution probability density function, the right figure shows the normal distribution cumulative probability function.

4 Reference

References

[1] John C. Hull. chapter 15 of *Options*, *Futures and Other Derivatives*, 9th Edition, Pearson Education.