BSM formula

Abstract

- create GBM class
- define a method for BSM formula for a given option type

Anal

BS model assumes the distribution of stock as lognormal. In particular, it writes

$$\ln \frac{S(T)}{S(0)} \sim \mathcal{N}((r - \frac{1}{2}\sigma^2)T, \sigma^2 T)$$

with respect to risk neutral measure. In the above, the parameters stand for

- $S(0)$: The initial stock price
- $S(T)$: The stock price at T
- r: interest rate
- σ: volatility

The call and put price with maturity T and K will be known as C_0 and P_0 given as below:

$$C_0 = \mathbb{E}[e^{-rT}(S(T) - K)^+] = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

and

$$P_0 = \mathbb{E}[e^{-rT}(S(T) - K)^-] = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1),$$

where d_i are given as

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

and

$$d_2 = \frac{(r - \frac{1}{2}\sigma^2)T - \ln \frac{K}{S_0}}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

Put-call parity will be useful:

$$C_0 - P_0 = S(0) - e^{-rT}K.$$

Code

```
import numpy as np
import scipy.stats as ss
```

We reload the european option class created before.

```
'''==========
option class init
==========='''
class EuropeanOption:
    def __init__(self,
                 otype = 1, # 1: 'call'
                            # -1: 'put'
                 strike = 110.,
```

```
                 maturity = 1.
                 ):
        self.otype = otype
        self.strike = strike
        self.maturity = maturity

    def payoff(self, s): #s: excercise price
        otype = self.otype
        k = self.strike
        maturity = self.maturity
        return np.max([0, (s - k)*otype])
```

Next, we create the gbm class, which is determined by three parameters. We shall initialize it as it is created.

```
'''============
Gbm class inherited from sde_1d
============='''

class Gbm:
    def __init__(self,
                 init_state = 100.,
                 drift_ratio = .0475,
                 vol_ratio = .2
                ):
        self.init_state = init_state
        self.drift_ratio = drift_ratio
        self.vol_ratio = vol_ratio
```

BSM formula is given by a method of Gbm class with an input of an option.

```
'''========
Black-Scholes-Merton formula.
==========='''

def bsm_price(self, european_option):
    s0 = self.init_state
    sigma = self.vol_ratio
    r = self.drift_ratio

    otype = european_option.otype
    k = european_option.strike
    maturity = european_option.maturity

    d1 = (np.log(s0 / k) + (r + 0.5 * sigma ** 2)
          * maturity) / (sigma * np.sqrt(maturity))
    d2 = d1 - sigma * np.sqrt(maturity)

    return (otype * s0 * ss.norm.cdf(otype * d1) #line break needs parenthesis
            - otype * np.exp(-r * maturity) * k * ss.norm.cdf(otype * d2))

Gbm.bsm_price = bsm_price
```

```
'''===============
Test bsm_price
================='''
gbm1 = Gbm()
option1 = EuropeanOption()
print('>>>>>>>>>>call value is ' + str(gbm1.bsm_price(option1)))
option2 = EuropeanOption(otype=-1)
print('>>>>>>>>>>put value is ' + str(gbm1.bsm_price(option2)))
```