BlackSholes

December 13, 2017

```
In [11]: %matplotlib inline
         import numpy as np
         import random
         import math
         import matplotlib.pyplot as plt

         def BlackSholes(drift, vola, init, t, rand):
             S_t = init * math.exp((drift - vola**2 / 2) * t + vola * math.sqrt(t) * rand )
             return S_t

         drift = 0.3
         vola = 0.3
         maturity = 1 # unit: year
         NumberOfPath = 100000
         StockPrice = np.zeros(NumberOfPath)
         StockPrice[0] = 100
         delta_t = maturity / NumberOfPath
         for i in range(1, NumberOfPath):
             StockPrice[i] = BlackSholes(drift, vola, StockPrice[i-1], delta_t, random.gauss(0,1

         #plt.plot(StockPrice)
         #Result = sum()
         #print(StockPrice)
```

```
In [144]: #TODO make class for returning process of Stock Price.
          #TODO write code for deriving call option price in the balck-sholes model.
```

1 12/13

1.1 Progress

Make GetProcess_Black - asset is assumed as stock - Inputs are initial value(init), drift term(drift), maturity of a transaction (assuming option pricing), number of path(num_of_path) - Outputs are given by the list of stockprice of each grid time. - Grid time are determined by the number of path and maturity

Make DrawPath_Black - This function is used for visializing the result of GetProcess_Black - Input are the same as GetProcess_Black except simulation number(simnum) - Simulation number corresponds to the number of path of one process

1.2 TODO クラス化して，stockpricess を保持できるようなクラスを作りたい（ざっくりベース)

- .get_process()
- .drawpath()
- みたいなメソッドを作りたい

```
In [2]: %matplotlib inline
        import numpy as np
        import random, math
        import matplotlib.pyplot as plt

        def GetProcess_Black(init, drift, volatility, maturity, num_of_path):
            delta_t = maturity / num_of_path
            stockprice = np.zeros(num_of_path)
            stockprice[0] = init
            for i in range(1, num_of_path):
                stockprice[i] = stockprice[i-1] * math.exp( (drift - ( volatility ** 2 / 2 ) ) *
                                                           delta_t + volatility * math.sqrt(delt
            return stockprice

        GetProcess_Black(100, 0.1, 0.5, 5, 100000)
        #plt.plot(GetProcess_Black(100, 0.1, 0.5, 5, 100000))

Out[2]: array([ 100.        ,  100.08984075,  100.25345136, ...,  235.8469375 ,
                235.42643021,  234.31372327])

In [48]: def DrawPath_Black(init, drift, volatility, maturity, num_of_path, simnum):
             #make list of list. If a dimentsion would be added, the same logic would be applica
             StockProcess = np.zeros(simnum*num_of_path).reshape(simnum,num_of_path)
             for i in range(simnum):
                 StockProcess[i] = list(GetProcess_Black(init, drift, volatility,
                                                         maturity, num_of_path))
             for i in range(simnum):
                 plt.plot(StockProcess[i])

In [57]: DrawPath_Black(100, 0.01, 0.1, 5, 100, 100)
```