Black Scholes and Monte Carlo Simulation

by Raul Maldonado

Table of Contents

1. Concepts and Procedure
   a. European Option Example
      i. Background
      ii. Concept
      iii. Calculation
   b. Monte Carlo Simulation Example
      i. Background
      ii. Feature Selection
      iii. Monte Carlo Method Example
2. Resources

Concepts and Procedure

Example One [3]

Background

Monte Carlo Method of credit portfolios is a probabilistic computation method typically used for the calculation of Credit Value at Risk and economic capital for credit portfolios. We approximate the loss distribution and estiamte various risk measures for credit portfolio.

Concept

We create a Monte Carlo simulation of the valuation of a European call option. We consider the Black Scholes Model.

Suppose we have the following *parameters values* for the valuation:

- Initial stock index level $S_0 = 100$
- Strike price of the European call option $K = 105$
- Time to maturity T = 1 year
- Systematic risk $r = 5\%$
- Constant volatility $\sigma = 20\%$

In the BSM model, the index level at maturity is a random variable, given by Equation 1-1 with z being a standard normally distributed random variable.

$$S_T = S_0 \exp\left((r - \frac{1}{2}\sigma^2)T + \sigma\sqrt{T}z\right)$$

The following is an algorithmic description of the Monte Carlo procedure:

1. Draw I (psuedo) random numbers $z(i), i \in \{1, 2, \ldots, I\}$, from the standard normal distribution
2. Calculate all resulting index levels at maturity $S_T(i)$ for given $z(i)$ and the Black Scholes Model $S_T = S_0 \exp\left((r - \frac{1}{2}\sigma^2)T + \sigma\sqrt{T}z\right)$
3. Calculate all inner values of the option at maturity as $h_T(i) = \max\left(S_T(i)K, 0\right)$
4. Estimate the option present value via Monte Carlo estimator below:
$$C_0 = e^{-rT}\frac{1}{I}\sum_{I} h_T(i)$$

where:

- S_t is the spot price of the underlying asset
- S_0 initial stock index
- k is the strike price
- $h_T(i)$ is our option at maturity

Calculation

In [13]:
```
import numpy as np
import random
import math
```

In [1]:
```
S0 = 100 # Intial stock index level
K = 105 # strike price
T = 1.0 # Time T (by years)
r = 0.05 # systematic risk
sigma = 0.2 #volatility
```

Import libraries and declare variables

In [24]:
```
sim = 100000 # number of simulations


z = np.random.standard_normal(sim)
# attain array of random numbers around
# the standard normal
```

Draw I (psuedo) random numbers $z(i), i \in \{1, 2, \ldots, I\}$, from the standard normal distribution

In [25]:
```
ST = S0 * np.exp((r - 0.5 * (sigma)**(2)) * T
                 + sigma * np.sqrt(T) * z)
#index values at maturity
```

Calculate all resulting index levels at maturity $S_T(i)$ for given $z(i)$ and the Black Scholes Model

In [26]:
```
hT = np.maximum(ST - K, 0)
# inner values at maturity
```

Calculate all inner values of the option at maturity

In [27]:
```
C0 = np.exp(-r * T) * sum(hT) / sim

# monte carlo simulation
```

Estimate option present value

In [29]:
```
print("Value of the European Call Option %5.3f" % C0)
```

```
Value of the European Call Option 7.987
```

Monte Carlo Method Example 2 [4]

Background

Systematic risk can be captured with liquidity index and specific risk can be modeled in regression models. Investment firms are required to hold capital to ensure solvency. The amount of capital that must be held is often calculated by risk models.

We validate risk models can be back-tested for goodness of fit using regression. The price-forecasting regression model can fail to capture the true market volatility. In this situation, we implement the regression model for the number of back-testing exceptions over a time-period

$$I_t = \beta_0 + \beta_1 X_{1,t-1} + \ldots + \beta_k X_{k,t-1} + \epsilon_t$$

where I_t is the total exceptions at time t, β_0 is a small acceptable number of exceptions at some confidence level, and betas are the weighted market volatility indices.

The back test would begin with a null hypothesis $H_0 : \beta_1 = \beta_2 = \ldots = \beta_k = 0$, and an alternative test to suppoort or reject it.

Features Selection

Features Selection is an important part of predictive modeling. Features compression is often done using Principle Component Analysis (PCA).

An investment portfolio of bonds with future cash flows is sensitive to changes in interest rates for different maturities. If we desire to estimate portfolio risk using a smaller number of factors, we can use PCA. By performing CPA on historical interest rate moves for set of maturities, we can select top first n factors explaining most of variation in data, where PCA is performed using a covariance matrix of short rate moves.

> We have been quoting [4] heavily, but here is somethign important to the following section:
>
> "One popular method for computing the value-at-risk is through revaluation of a portfolio under a set of moves in price and volatility. When dealing with derivatives that are so-called ‘path dependent’ (e.g. Asian options) one needs to consider entire volatility term structure as opposed to a volatility at a single tenor. However, applying moves to all volatilities is computationally expensive. One solution is to use the output of PCA to reduce the term-structure."

Monte Carlo Method Example

Reminding ourselves, Monte Carlo Method is used extensively in quantitative finance. The dynamics of a Monte Carlo Simultation is assumed to follow stochastic process, like **Geometric Brownian Motion**.

The simulated prices become inputs into a payoff function, and the avg(discounted payoff) determines the price of derivative.

In [34]:
```
import numpy as np
from scipy.stats import norm

r = 0.01       #risk -free rate
sigma = 0.05   # volatility
K = 18.0       # Strike Price (K)
S0 = 18.0      # Intiial Stock value at time t=0
T = 10 # Time till maturity
```

Import libraries and establish variables

In [35]:
```
# Black Scholes Model Implementation

disc = np.exp(- r* T)     #Discounting: e^(-rt)
vol = sigma * np.sqrt(T)  # time-scaled volatility (sigma * sqrt(T))

d1 = ((np.log(S0/K)) + ((r + 0.5 * sigma**2) *T)) / vol
d2 = d1 - vol

black_scholes_call_price = S0 * norm.cdf(d1) - disc * K * norm. cdf(d2)
```

$$\text{Black Scholes Model} : C = S_0 N(d_1) - Ke^{-rt} N(d_2)$$

Above, we implement the Black Scholes Model for future comparison with that of the Monte Carlo Method.

In [36]:
```
# Simulation
N = 10000000      #Number of simulations
np.random.seed(0)
rands = np.random.normal(size = N)
```

Intiate standard distribution of random normal distribution numbers

In [41]:
```
ST = S0 * np.exp((r- 0.5 * sigma**2) *T + vol * rands)
payoff = np.multiply ([p if p >0 else 0 for p in ST-K], disc)
MCprice = np.mean(payoff, axis = 0)
```

$$C = S_0 e^{r - \frac{\sigma^2}{2} T + \sigma\sqrt{T} w_t}$$

Above is the implemenation of Monte Carlo Simulation

where $\frac{\sigma^2}{2} T$ is our drift and random stochiastic process is $\sigma\sqrt{T} w_t$

In [38]:
```
print("Option price with BS and MC are : %f and %f" % (black_scholes_call_pric
e, MCprice))
```

```
Option price with BS and MC are : 2.145200 and 2.145654
```

We observe a close margin of predicting some call price $C = 0.000454$ between the Black Schole model and the Monte Carlo Simulation

Resources

Document Resources

1. Black Scholes Model (https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model)
2. Monte Carlo Simulation (http://www.maths-in-industry.org/miis/172/1/algorithmics.pdf)

   a. Understanding Monte Carlo Simulation (https://www.youtube.com/watch?v=3gcLRU24-w0)
3. Monte Carlo Example, from above (https://www.safaribooksonline.com/library/view/python-for-finance/9781491945360/ch03.html)
4. Advanced Monte Carlo simulation example (http://www.kdnuggets.com/2016/12/data-analytics-models-quantitative-finance-risk-management.html)

Additional Readings

1. Counter Party Credit Risk book that I want to read (http://radoudoux.free.fr/last2/jGregoryCPTY%20Risk.pdf)

Hopefully you enjoyed my quick summarization of the Monte Carlo Method!

Thank you for reading!