Lecture 9: BS Model and Implied Vol

Cheng Lu

Overview

- Black-Scholes Model
- Implied Volatility
  - Implied Volatility for Market Data
  - Implied Volatility for All Expiration Dates

Black-Scholes Model

Black-Scholes Model is a mathematical model for pricing financial instruments (or financial derivatives). The main assumption is that the stock price follows geometric Brownian Motion

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), S(0) = S_0$$

where μ is a constant drift, σ is a constant volatility, and $W(t)$ is a Brownian Motion. The risk neutral dynamic is given by

$$dS(t) = rS(t)dt + \sigma S(t)dW^Q(t), S(0) = S_0$$

where r is the risk free rate (or interest rate).

Black-Scholes Model

The payoff of the a call option[1] is

$$(S(T) - K)_+ = \max\{S(T) - K, 0\}$$

Then the risk neutral pricing formula implies the call option price c at time 0 is given by

$$c(S_0, K, T, \sigma, r) = e^{-rT}E^Q[(S(T) - K)_+] = S_0 N(d_1) - e^{-rT}KN(d_2)$$

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

where $N(\cdot)$ is the cumulative distribution function of standard normal distribution (i.e. mean $= 0$, sd $= 1$). And dividend is not considered here.

[1] Recall a call option is a contract that the investor has right to buy the underlying stock at specific time T with specific price K. Here T is called maturity (or expiration date, expiry date) and K is called strike price.

Black-Scholes Model

Example

```
> S0 <- 100 # spot price
> K <- 100 # strike price
> T1 <- 1 # maturity
> # if we let T <- 1, "T" would not be "TRUE" anymore
> sigma <- 0.2 # volatility
> r <- 0.05 # risk free rate
>
> d1 <- (log(S0/K) + (r+0.5*sigma^2)*T1)/(sigma*sqrt(T1))
> d2 <- d1 - sigma*sqrt(T1)
> (c <- S0*pnorm(d1) - exp(-r*T1)*K*pnorm(d2))
[1] 10.45058
```

Black-Sccholes Model

Then we can write it into a function

Example

```
> bs.call <- function(S0, K, T1, sigma, r){
+   d1 <- (log(S0/K) + (r+0.5*sigma^2)*T1)/(sigma*sqrt(T1))
+   d2 <- d1 - sigma*sqrt(T1)
+   S0*pnorm(d1) - exp(-r*T1)*K*pnorm(d2)
+   # return(S0*pnorm(d1) - exp(-r*T1)*K*pnorm(d2))
+ }
> bs.call(S0, K, T1, sigma, r) # bs.call(S0, K, T1, r, sigma) doesn't work
[1] 10.45058
> bs.call(100, 100, 1, r = 0.05, sigma = 0.2)
[1] 10.45058
> bs.call(100, 100, 1, 0.2, 0.05)
[1] 10.45058
```

Implied Volatility

Implied volatility is the volatility σ such that the price from Black-Scholes model equals to the price from market. So, for call option we need to find a σ such that

$$c(S_0, K, T, \sigma, r) = P \Leftrightarrow c(S_0, K, T, \sigma, r) - P = 0$$

where P denote the price from market.

- We can use the root finding methods (e.g. bisection method in L8) to find σ.
- If the market price of the call option is $P = 10$, we can find σ between 0 and 1.[2]

Example

```
> price.diff <- function(sigma)bs.call(S0,K,T1,sigma,r)-10
> bisection.new(price.diff,0,1)
[1] 0.1879883
```

where the **bisection.new()** is defined in L8. Build-in function **uniroot** also works.

[2] Sometime σ calculated from real market data may grater than 1, we may need larger value in practice.

Implied Volatility

If we want to use Newton-Raphson method (in L8), we need the partial derivative of the call option price with respect to σ, which is also called Vega

$$Vega = \frac{\partial c}{\partial \sigma} = \sqrt{T} S_0 N'(d_1); \, d_1 = \frac{\ln \frac{S_0}{K} + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}$$

where $N'(\cdot)$ is the derivative of $N(\cdot)$, which is the density of the standard normal distribution.

Example

```
> Vega <- function(S0, K, T1, sigma, r){
+   d1 <- (log(S0/K) + (r+0.5*sigma^2)*T1)/(sigma*sqrt(T1))
+   sqrt(T1)*S0*dnorm(d1)
+ }
> dprice.diff <- function(sigma)Vega(S0,K,T1,sigma,r)
> Newton_Raphson(price.diff,dprice.diff,0.25)
[1] 0.1879716
```

Implied Volatility

Next step is to write a function to calculate implied volatility:

Example

```
> implied.vol.call <- function(S0, K, T1, r, price, method="bisection"){
+   price.diff <- function(sigma)bs.call(S0, K, T1, sigma, r) - price
+   if(method == "bisection"){
+     return(bisection.new(price.diff, 0.01, 5))
+   }else if(method == "Newton-Raphson"){
+     dprice.diff <- function(sigma)Vega(S0, K, T1, sigma, r)
+     return(Newton_Raphson(price.diff, dprice.diff, 0.25))
+   }
+ }
> implied.vol.call(S0,K,T1,r,10)
[1] 0.1879883
> implied.vol.call(S0,K,T1,r,10,"Newton-Raphson")
[1] 0.1879716
```

- **bs.call()** is on page 6 and **Vega()** is on page 8
- **bisection.new()** and **Newton_Raphson()** are from L8.

Implied Volatility for Market Data

When we calculate implied volatility with market data (e.g. SPY options), we need to determine S_0, K, T, r, and P.

- Spot price S_0 is the current price of the underlying, we can use the last quoted price

Example

```
> library(quantmod)
> (SPY.S0 <- getQuote("SPY")$Last)
[1] 400.64
```

- Risk free rate r is the current yield form the risk free assets
  - We can use the rates from Federal Reserve website[3] (e.g. federal funds effective rate)
  - The rate from the website is in percentage, we need to multiply 0.01

Example

```
r <- 0.07 * 0.01
```

[3] http://www.federalreserve.gov/releases/h15

Implied Volatility for Market Data

- Strike price K time and maturity T are written on contracts
- Price P can be measured by the average of bid price and ask price

Example

```
> SPY.option <- getOptionChain("SPY")
> head(SPY.option$calls)# nearest maturity option
                   Strike   Last Chg    Bid    Ask Vol  OI       LastTradeTime       IV  ITM
SPY210405C00240000    240 145.56   0 160.78 161.21  NA  74 2021-03-25 11:51:12 3.152346 TRUE
SPY210405C00245000    245 141.36   0 155.78 156.21  NA  98 2021-03-25 11:56:42 3.037112 TRUE
SPY210405C00250000    250 136.33   0 150.78 151.21  NA 147 2021-03-25 10:29:07 2.925784 TRUE
SPY210405C00255000    255 141.12   0 145.78 146.21  20 171 2021-03-31 09:43:10 2.814456 TRUE
SPY210405C00260000    260 126.18   0 140.78 141.21  NA 120 2021-03-25 10:17:16 2.707034 TRUE
SPY210405C00265000    265 121.17   0 135.77 136.21  NA  96 2021-03-25 09:57:20 2.595707 TRUE
```

The name SPY210405C00240000 means

- Ticker is "SPY"; Maturity is 2021-04-05
- Type Call "C"; Strike price K is \$240.0

Implied Volatility for Market Data

If we want to calculate the implied volatility for the option in the first row, then

Example

```
> (SPY.K1 <- SPY.option$calls$Strike[1])
[1] 240
> (SPY.Price1 <- 0.5*(SPY.option$calls$Bid[1] + SPY.option$calls$Ask[1]))
[1] 160.995
> (T1 <- as.numeric(as.Date("2021-04-05") - Sys.Date())/252)
[1] 0.003968254
```

Implied Volatility for Market Data

With the parameter given above, we can calculate the implied volatility using the function defined on page 9

Example

```
> implied.vol.call(SPY.S0,SPY.K1,T1,r,SPY.Price1)
[1] 3.668745
```

This output may not be good. Because

- Expiration date is too close (1 day to maturity)
- Option is deep in the money.
- Less people trade the option, price may not reflect its value

Example

```
> bs.call(SPY.S0, SPY.K1, T1, 0, r)-SPY.Price1
[1] -0.3311917
> bs.call(SPY.S0, SPY.K1, T1, 5, r)-SPY.Price1
[1] 1.777545
```

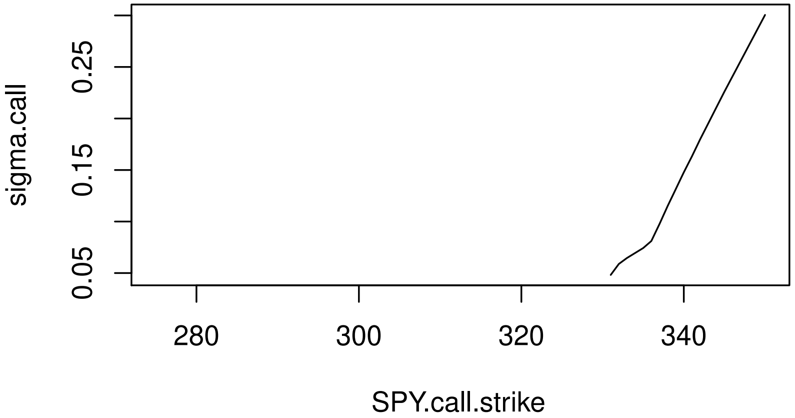
Implied Volatility for Market Data

- In case of the root doesn't exists, we may get a warning or error to stop the algorithm
- Need to modify the algorithm to return NA when $f(a) * f(b)$ is NA or positive (See exercise 1 in L8).

Example

```
> SPY.call.strike <- SPY.option$calls$Strike
> SPY.call.price <- 0.5*(SPY.option$calls$Bid+SPY.option$calls$Ask)
> sigma.call <- NULL
> for (i in 1:nrow(SPY.option$calls)){
+   sigma.call[i] <- implied.vol.call(SPY.S0,SPY.call.strike[i],
+                                     T1, r, SPY.call.price[i])
+ }
> plot(SPY.call.strike, sigma.call, type = "l")
```

Implied Volatility for Market Data



Modified Bisection Method

Here is the modified bisection method from the last lecture as the reference

Example: Modified Bisection Method

```
bisection.new <- function(f, a, b, tol = 0.001, N.max = 100){
  f.a <- f(a)
  f.b <- f(b)
  if(is.na(f.a*f.b) || f.a*f.b > 0){# only modified this part
    return(NA)
  }else if(f.a == 0){
    return(a)
  }else if(f.b == 0){
    return(b)
  }
```

Modified Bisection Method

Example: Modified Bisection Method Continued

```
  for(n in 1:N.max){
    c <- (a+b)/2
    f.c <- f(c)
    if(f.c == 0 || abs(b - a) < tol){
      break
    }
    if(f.a*f.c < 0){
      b <- c
      f.b <- f.c
    }else{
      a <- c
      f.a <- f.c
    }
  }
  return(c)
}
```

Implied Volatility for All Expiration Dates

Next we calculate implied volatility for *all* call options by adding columns of implied volatilities to the sub-lists.

Example

```
> SPY.options.all <- getOptionChain("SPY", NULL)# all options
> SPY.S0 <- getQuote("SPY")$Last
> r <- 0.07 * 0.01
> SPY.expiration <- names(SPY.options.all)# all expiration dates
> T.vec <- (as.Date(SPY.expiration,"%b.%d.%Y")-Sys.Date())/365# calendar day
> T.vec <- as.numeric(T.vec)# all time to maturities
```

Implied Volatility for All Expiration Dates

We can use loops to calculate the implied volatilities for call options at each expiration:

Example

```
for(i in 1:length(SPY.options.all)){
  SPY.options.all[[i]]$calls$Price <- 0.5*(SPY.options.all[[i]]$calls$Bid
                                         + SPY.options.all[[i]]$calls$Ask)
  for(j in 1:nrow(SPY.options.all[[i]]$calls)){
    SPY.options.all[[i]]$calls$impliedVol[j] <-
      implied.vol.call(SPY.S0,
                       SPY.options.all[[i]]$calls$Strike[j],
                       T.vec[i],
                       r,
                       SPY.options.all[[i]]$calls$Price[j])
  }
  SPY.options.all[[i]]$calls <-
    SPY.options.all[[i]]$calls[c("Bid", "Ask", "Strike","Price","impliedVol")]
}
```

Here **implied.vol.call()** function is defined on page 9.

Implied Volatility for All Expiration Dates

If we use vectorized calculation instead of loops, we have

Example

```
SPY.options.all <- getOptionChain("SPY", NULL)# all options
calc <- function(x, T1){
  # add a column of price
  x$calls$Price <- 0.5*(x$calls$Bid + x$calls$Ask)
  # add a column of implied volatility
  func <- function(K,P)implied.vol.call(SPY.S0,K,T1,r,P)
  x$calls$impliedVol <- mapply(func, x$calls$Strike, x$calls$Price)
  # delete columns
  x$calls <- x$calls[c("Bid","Ask","Strike","Price","impliedVol")]
  return(x)
}
SPY.options.all <- mapply(calc, SPY.options.all, T.vec, SIMPLIFY = FALSE)
```

Implied Volatility for All Expiration Dates

Then we can choose 3 maturities to plot implied volatility versus strike price.

Example

```
# Initializing figure manually
plot(NA, xlim = c(300,500), ylim = c(0,0.4), xlab = "Strike",
     ylab = "ImpliedVol") # xlim: range of strike, ylim: range of vol
# Add lines
lines(SPY.options.all[[14]]$calls$Strike,
      SPY.options.all[[14]]$calls$impliedVol,col = "red")
lines(SPY.options.all[[19]]$calls$Strike,
      SPY.options.all[[19]]$calls$impliedVol,col = "green")
lines(SPY.options.all[[21]]$calls$Strike,
      SPY.options.all[[21]]$calls$impliedVol,col = "blue")
legend("bottomleft", SPY.expiration[c(14,19,21)],
       fill = c("red","green","blue"))
```

Implied Volatility for All Expiration Dates