MSPA PREDICT 400

Discussion Topic: Week 6 Derivatives

Introduction

This document presents the results of the sixth w eeks discussion topic for the Masters of Science in Predictive Analytics course: PREDICT 400. This assessment required the student to explain w hat a derivate is, and how they have played a role in their life.

Black-Scholes Option Pricing Formula

Since my undergraduate studies w ere in Finance, I decided to take a look at the various option greeks.

The Black-Scholes option pricing formula below is relevant for valuation of European call/ put options.

$$C(S,t) = N(d_1)S - N(d_2)Ke^{-r(T-t)}$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}\left[ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right]$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

w here:

C = Call premium

S = Current price of the underlying

T = Time of option exercise

t = Time of option valuation

K = Option strike price

r = Risk-free rate

σ = Price volatility of the underlying

Partial Derivatives

There are a number of importand partial derivatives of the option pricing formula show n above. One of w hich is the delta of the option price.

Delta measures the rate of change of the theoretical option value w ith respect to changes in the underlying asset's price. It is the first derivate of the value of the option (V), w ith respect to the underlying instruments price (S).

$$delta = \frac{V(S+dS) - V(S)}{dS}$$

For example, w ith respect to call options, a delta of 0.6 w ould mean that for every $1.00 increase in the price of the underlying, the call option value w ould increase by $0.60.

Visualizing Option Greeks

To demonstrate the delta visually, I have leveraged Python code from here, in order to 1) plot the value of a call option w .r.t changes in the spot price of the underlying, and 2) plot delta of the same call option w .r.t changes in the spot price of the underlying.

Note that the orignally sourced Python code reference above generates plots w .r.t changes in the strike price of the option. I have changed relevant code so that the plots below are w .r.t changes in the spot price of the underlying. Also note, the index parameter values for each of the option paramters below :

St = 100.0 # index level

K = 100.0 # option strike

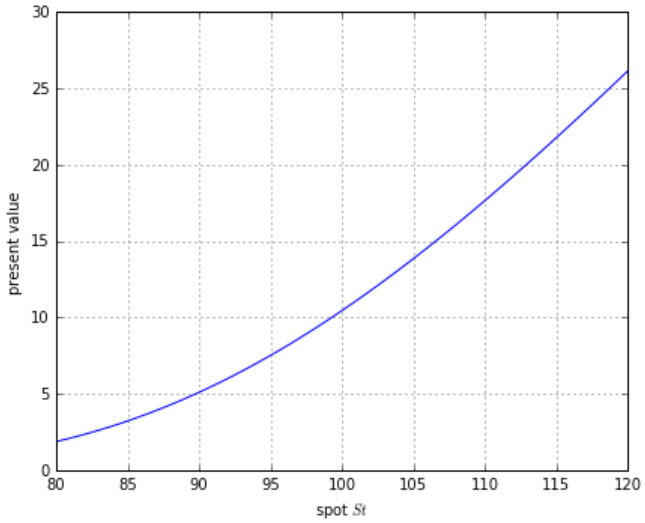
T-t = 1.0 # maturity date to valuation date

r = 0.05 # risk-less short rate

sigma = 0.2 # volatility

In [1]:
```
#%run BSM_option_valuation.py
#%run BSM_call_greeks.py
```

In [2]:
```
#Source: https://github.com/yhilpisch/dawp

import math
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.integrate import quad
%matplotlib inline

def dN(x):
    return math.exp(-0.5 * x ** 2) / math.sqrt(2 * math.pi)

def N(d):
    return quad(lambda x: dN(x), -20, d, limit=50)[0]

def d1f(St, K, t, T, r, sigma):
    d1 = (math.log(St / K) + (r + 0.5 * sigma ** 2)
            * (T - t)) / (sigma * math.sqrt(T - t))
    return d1

def plot_values(function):
    plt.figure(figsize=(15, 12))
    points = 100
    St = 100.0  # index level
    K = 100.0  # option strike
    t = 0.0  # valuation date
    T = 1.0  # maturity date
    r = 0.05  # risk-less short rate
    sigma = 0.2  # volatility

    # C(St) plot
    plt.subplot(221)
    klist = np.linspace(80, 120, points)
    vlist = [function(St, K, t, T, r, sigma) for St in klist]
    plt.plot(klist, vlist)
    plt.grid()
    plt.xlabel('spot $St$')
    plt.ylabel('present value')

def BSM_call_value(St, K, t, T, r, sigma):
    d1 = d1f(St, K, t, T, r, sigma)
    d2 = d1 - sigma * math.sqrt(T - t)
    call_value = St * N(d1) - math.exp(-r * (T - t)) * K * N(d2)
    return call_value

plot_values(BSM_call_value)
```

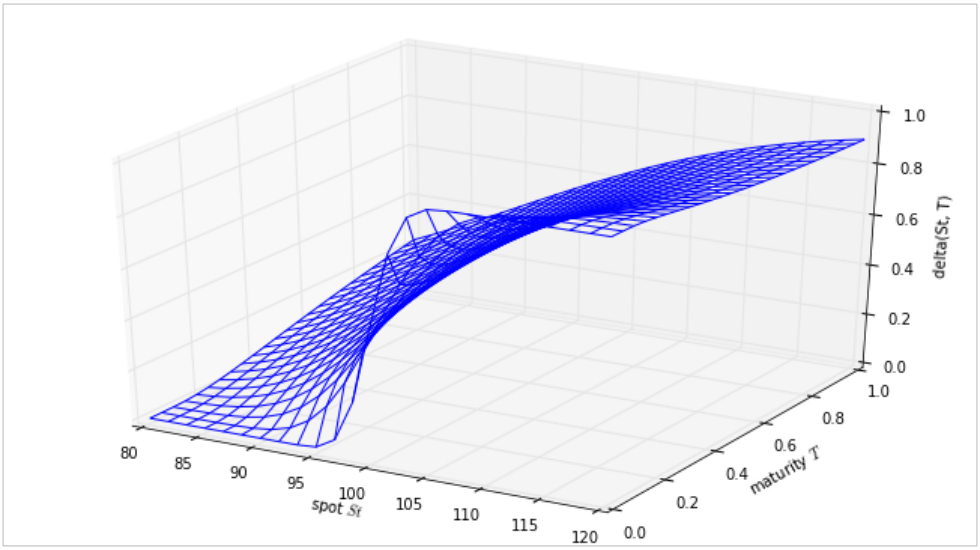


It can be seen in the plot above that as the spot price of the underlying increase, the call option value also increases. This is intuitive as the call option becomes more in the money as the spot price of the underlying advances beyond the strike price and more out of the money as the spot price of the underling falls below the strike price.

How ever, note that the response of change in option value w .r.t the spot price of the underlying is not linear. To see this in more detail, examine the face of the plot below w hich show s delta w .r.t to the spot price of the underlying.

In [3]:
```
import math
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.axes3d as p3
%matplotlib inline

def dN(x):
    return math.exp(-0.5 * x ** 2) / math.sqrt(2 * math.pi)

def N(d):
    return quad(lambda x: dN(x), -20, d, limit=50)[0]

def d1f(St, K, t, T, r, sigma):
    d1 = (math.log(St / K) + (r + 0.5 * sigma ** 2)
            * (T - t)) / (sigma * math.sqrt(T - t))
    return d1

def plot_greeks(function, greek):
    # Model Parameters
    St = 100.0  # index level
    K = 100.0  # option strike
    t = 0.0  # valuation date
    T = 1.0  # maturity date
    r = 0.05  # risk-less short rate
    sigma = 0.2  # volatility

    # Greek Calculations
    tlist = np.linspace(0.01, 1, 25)
    stlist = np.linspace(80, 120, 25)
    V = np.zeros((len(tlist), len(stlist)), dtype=np.float)
    for j in range(len(stlist)):
        for i in range(len(tlist)):
            V[i, j] = function(stlist[j], K, t, tlist[i], r, sigma)

    # 3D Plotting
    x, y = np.meshgrid(stlist, tlist)
    fig = plt.figure(figsize=(9, 5))
    plot = p3.Axes3D(fig)
    plot.plot_wireframe(x, y, V)
    plot.set_xlabel('spot $St$')
    plot.set_ylabel('maturity $T$')
    plot.set_zlabel('%s(St, T)' % greek)
```

In [4]:
```
def BSM_delta(St, K, t, T, r, sigma):
    d1 = d1f(St, K, t, T, r, sigma)
    delta = N(d1)
    return delta

plot_greeks(BSM_delta, 'delta')
```



As it can be seen above, delta increases as the spot price of the underlying increases, how ever, delta is much more responsive for changes in spot around the strike price.

To see this in more detail, w e can go one step further and plot gamma. Gamma is the second derivative of the option value w ith respect to the price of the underlying.

In [5]:
```
def BSM_gamma(St, K, t, T, r, sigma):
    d1 = d1f(St, K, t, T, r, sigma)
    gamma = dN(d1) / (St * sigma * math.sqrt(T - t))
    return gamma

plot_greeks(BSM_gamma, 'gamma')
```