martingale-theory-to-option-price

October 3, 2023

[4]:
```
import numpy as np
from scipy.stats import norm
```

Delta: Delta measures the sensitivity of an option's price to changes in the price of the underlying asset.

[4]:
```
def delta(option_type, S, K, r, T, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    if option_type == "call":
        return np.exp(-r * T) * norm.cdf(d1)
    elif option_type == "put":
        return -np.exp(-r * T) * norm.cdf(-d1)
```

[8]:
```
option_type = "call"
S = 100   # Current stock price
K = 95    # Strike price
r = 0.052  # Risk-free rate
T = 0.5   # Time to expiration (in years)
sigma = 0.2  # Volatility
```

[10]:
```
call_delta = delta(option_type, S, K, r, T, sigma)*100
print("Call Delta:", call_delta)
```

```
Call Delta: 71.26942799282314
```

Gamma

[11]:
```
def gamma(S, K, r, T, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    return np.exp(-r * T) * norm.pdf(d1) / (S * sigma * np.sqrt(T))

# Example usage for a call option
S = 100   # Current stock price
K = 95    # Strike price
r = 0.05  # Risk-free rate
T = 0.5   # Time to expiration (in years)
sigma = 0.2  # Volatility

call_gamma = gamma(S, K, r, T, sigma)*100
```

```
print("Call Gamma:", call_gamma)
```

```
Call Gamma: 2.2839574296269998
```

Theta: Theta measures the sensitivity of an option's price to changes in time (time decay).

[13]:
```
def theta(option_type, S, K, r, T, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    if option_type == "call":
        return (-S * norm.pdf(d1) * sigma / (2 * np.sqrt(T)) - r * K * np.
 ↪exp(-r * T) * norm.cdf(d2))
    elif option_type == "put":
        return (-S * norm.pdf(d1) * sigma / (2 * np.sqrt(T)) + r * K * np.
 ↪exp(-r * T) * norm.cdf(-d2))


    # Example usage for a call option
option_type = "call"
S = 100   # Current stock price
K = 95    # Strike price
r = 0.05  # Risk-free rate
T = 0.5   # Time to expiration (in years)
sigma = 0.2  # Volatility


call_theta = theta(option_type, S, K, r, T, sigma)
print("Call Theta:", call_theta)
```

```
Call Theta: -7.835568111084916
```

Vega measures the sensitivity of an option's price to changes in implied volatility.

[14]:
```
def vega(S, K, r, T, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    return S * np.exp(-r * T) * norm.pdf(d1) * np.sqrt(T)
```

[15]:
```
# Example usage for a call option
S = 100   # Current stock price
K = 95    # Strike price
r = 0.05  # Risk-free rate
T = 0.5   # Time to expiration (in years)
sigma = 0.2  # Volatility
```

[16]:
```
call_vega = vega(S, K, r, T, sigma)
print("Call Vega:", call_vega)
```

```
Call Vega: 22.839574296269998
```

RHO often denoted as , is one of the Greeks used in options trading to measure the sensitivity of an option's price to changes in the risk-free interest rate. In other words, rho quantifies how much an option's value changes for a one-unit change in the interest rate.

[1]:
```
def calculate_rho(option_type, S, K, r, T, sigma):
    d2 = (np.log(S / K) + (r - 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))

    if option_type == "call":
        rho = K * T * np.exp(-r * T) * norm.cdf(d2)
    elif option_type == "put":
        rho = -K * T * np.exp(-r * T) * norm.cdf(-d2)

    return rho
```

[2]:
```
# Example usage for a call option
option_type = "call"
S = 100   # Current stock price
K = 95    # Strike price
r = 0.05  # Risk-free rate
T = 0.5   # Time to expiration (in years)
sigma = 0.2  # Volatility
```

[5]:
```
call_rho = calculate_rho(option_type, S, K, r, T, sigma)
print("Call Rho:", call_rho)
```

```
Call Rho: 31.520159366235706
```

[6]:
```
# Example usage for a put option
option_type = "put"
put_rho = calculate_rho(option_type, S, K, r, T, sigma)
print("Put Rho:", put_rho)
```

```
Put Rho: -14.807061455110091
```