Extensions of Options Theory

資工三 B04902009 蕭千惠
30th April, 2018

Barrier Options

Knock-out (KO) option is an ordinary European optin which ceases to exist if the barrier H is reached by the price of its underlying asset.

Knock-in (KI) option comes into existence if a certain barrier is reached.

	call	put
knock-out	down-and-out $(H < S \leq \infty)$	up-and-out $(0 \leq S < H)$
knock-in	down-and-in $(0 \leq S < H)$	up-and-in $(H < S \leq \infty)$

The value of a European barrier options on a stock paying a dividend yield of q is

- Down-and-in call

$$Se^{-q\tau}\left(\frac{H}{S}\right)^{2\lambda}\mathrm{N}(x) - Xe^{-r\tau}\left(\frac{H}{S}\right)^{2\lambda-2}\mathrm{N}(x-\sigma\sqrt{\tau})$$

  where

$$x = \frac{\ln(H^2/SX) + (r-q+\sigma^2/2)\tau}{\sigma\sqrt{\tau}}$$

$$\lambda = \frac{(r-q+\sigma^2/2)}{\sigma^2}$$

- Down-and-out call

  Can be priced via the in-out parity.

- Up-and-in put

$$Xe^{-r\tau}\left(\frac{H}{S}\right)^{2\lambda-2}\mathrm{N}(-x+\sigma\sqrt{\tau}) - Se^{-q\tau}\left(\frac{H}{S}\right)^{2\lambda}\mathrm{N}(-x)$$

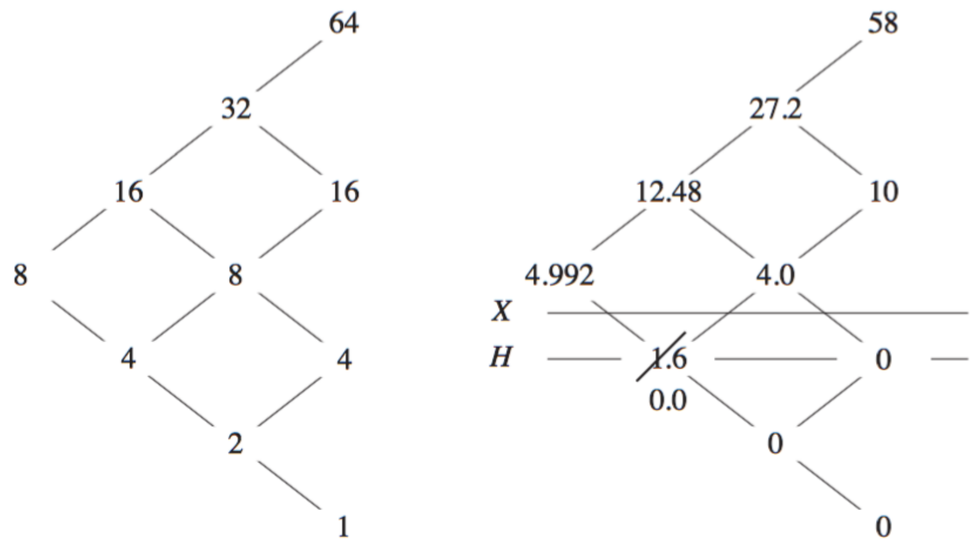
- Up-and-out put

  Can be priced via the in-out parity.

Binomial Tree Algorithms

Barrier options can be priced by binomial tree algorithms.

Example: Down-and-out



$S = 8$, $X = 6$, $H = 4$, $R = 1.25$, $u = 2$, and $d = 0.5$.
Backward-induction: $C = (0.5 \times C_u + 0.5 \times C_d)/1.25$.

backward reduction:碰到 H 這條線的點全部設爲 0，其餘不變，線以下的點都不用繼續算。

But convergence is erratic because H is not at a price level on the tree.
H 不一定會剛好落在 tree 的某個點上，在做 backward reduction 不得已把 H 往上或往下移到某個存在的股價上（edge 被移動）。不同的 N（# of periods），edge 都不一樣，等同於解不同的 barrier option，故收斂效果差。

Path-Dependent Derivatives

Its value depends only on the underlying asset's terminal price regardless of how it gets there.

Average-rate options: also called **Asian options**.

- Arithmetic average-rate call's terminal value: $\max\left(\frac{1}{n+1}\sum_{i=0}^{n} S_i - X, 0\right)$
- Arithmetic average-rate put's terminal value: $\max\left(X - \frac{1}{n+1}\sum_{i=0}^{n} S_i, 0\right)$

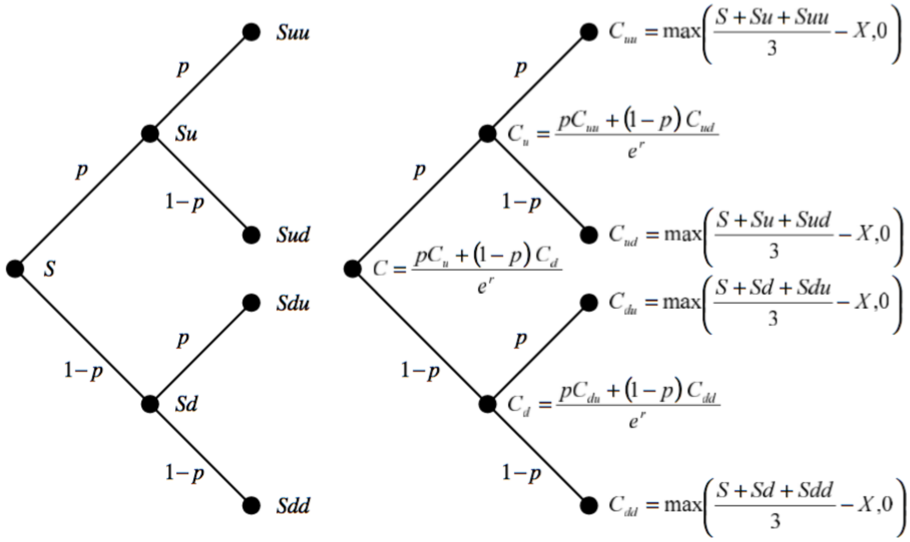
Lookback option: Strike price isn't fixed.

- Terminal payoff of lookback call option on the minimum: $S_n - \min_{0\le i\le n} S_i$
- Terminal payoff of lookback put option on the maximum: $\max_{0\le i\le n} S_i - S_n$
- Terminal payoff of fixed-strike lookback call option: $\max\left(\max_{0\le i\le n} S_i - X, 0\right)$
- Terminal payoff of fixed-strike lookback put option: $\max\left(X - \max_{0\le i\le n} S_i, 0\right)$

Average-strike options: lookback calls and puts on the average (instead of a constant X).

Average-Rate Options (Asian Options)

The binomial tree for the averages does not combine.



State for the Asian option: the tuple

$$(i, S, P)$$

where i is the time, S is the prevailing stock price, and P is the running sum.

For the binomial model, the state transition is:

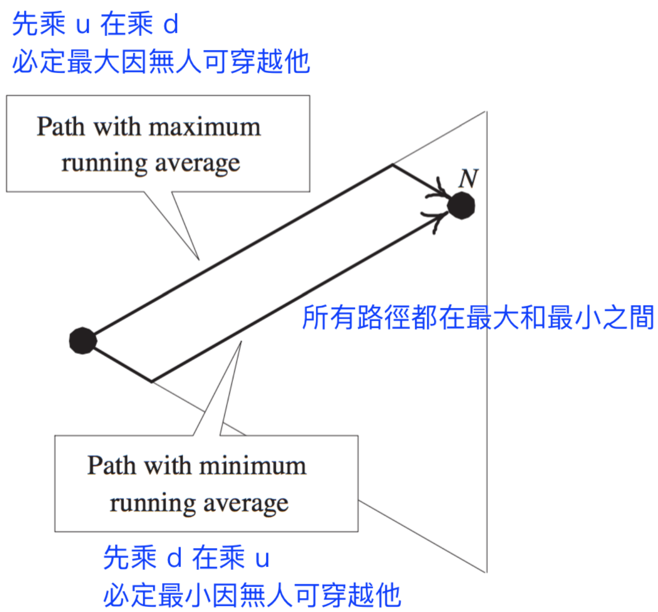
$$(i, S, P) \begin{array}{l} \nearrow \ (i+1, Su, P+Su), \text{ for the up move} \\ \\ \searrow \ (i+1, Sd, P+Sd), \text{ for the down move} \end{array}$$

Approximation Algorithm for Asian Options (Based on BOPM)

Step 1: Running Average

Let $N(j, i)$ denotes the node at time j with the underlying asset price equal to $S_0 u^{j-i} d^i$

The running sum at node $N(j, i) = \sum_{m=0}^{j} S_m$



- The running sum has a maximum value:

$$S_0(1 + u + u^2 + \ldots + u^{j-i} + u^{j-i}d + u^{j-i}d^i)$$

$$= S_0 \frac{1 - u^{j-i+1}}{1-u} + S_0 u^{j-i} d \frac{1-d^i}{1-d}$$

Running averages: divide this value by $j + 1$ and call it $A_{\max}(j, i)$.

- The running sum has a minimum value:

$$S_0(1 + d + d^2 + ... + d^{j-i} + d^{j-i}u + d^{j-i}u^i)$$
$$= S_0 \frac{1 - d^{i+1}}{1 - d} + S_0 d^i u \frac{1 - u^{j-i}}{1 - u}$$

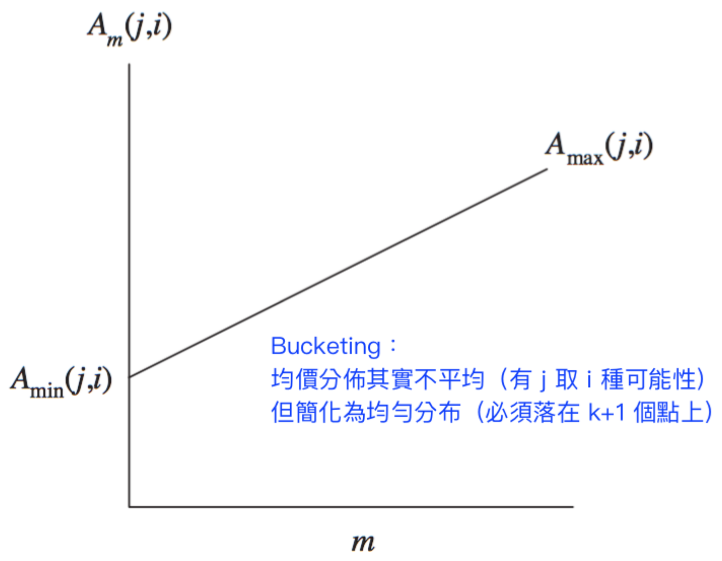
Running averages: divide this value by $j + 1$ and call it $A_{\min}(j, i)$.

All averages must lie between $A_{\min(j,i)}$ and $A_{\max(j,i)}$

Step 2: Bucketing

Pick $k + 1$ equally spaced values in range $[A_{\min(j,i)}, A_{\max(j,i)}]$ and treat them as the true and only running averages. For $m = 0, 1, ..., k$

$$A_m(j, i) = \left(\frac{k - m}{k}\right) A_{\min}(j, i) + \left(\frac{m}{k}\right) A_{\max}(j, i)$$



Bucketing introduces errors, but it works reasonably well in practice.

A better alternative picks values whose logarithms are equally spaced.

Step 3: Backward induction

Calculates the option values at each node for the $k+1$ running averages.

Suppose the current node is $N(j,i)$ and the running average is a.

Assume the next node is $N(j+1,i)$ after an up move.

1. Calculate A_u

   Asset price: $S_0 u^{j+1-i} d^i$

   New running average:

   $$A_u = \frac{(j+1)a + S_0 u^{j+1-i} d^i}{j+2}$$

2. Find l

   Since A_u is not likely to be one of the $k+1$ running averages at $N(j+1,i)$, it's required to find the 2 running average that bracket A_u:

   $$A_l(j+1,i) \le A_u < A_{l+1}(j+1,i)$$

   In "most" cases, the fastest way to nail l is via:

   $$l = \left\lfloor \frac{A_u - A_{\min}(j+1,i)}{[A_{\max}(j+1,i) - A_{\min}(j+1,i)]/k} \right\rfloor$$

   But watch out for some rare case,

   - $A_u = A_l(j+1,i)$ for some l
   - $A_u = A_{\max}(j+1,i)$
   - $A_0(j+1,i) = ... = A_k(j+1,i)$, which happen along extreme paths

3. Find x

   Express A_u as a linearly interpolated value of the two running averages:

   $$A_u = xA_l(j+1,i) + (1-x)A_{l+1}(j+1,i), 0 < x \le 1$$

4. Calculate C_u

   Obtain the approximate option value given the running average A_u via

   $$C_u = xC_l(j+1,i) + (1-x)C_{l+1}(j+1,i)$$

   where $C_l(t,s)$ denotes the option value at node $N(t,s)$ with running average $A_l(t,s)$.

The same steps are repeated for the down node $N(j+1, i+1)$ to obtain another approximate option value C_d.

Finally obtain the option value

$$[pC_u + (1-p)C_d]e^{-r\Delta t}$$

For the calculations at time step $n-1$, no interpolation is needed.

The option values for calls are simply

$$C_u = \max(A_u - X, 0)$$

$$C_d = \max(A_d - X, 0)$$

Remark on Asian Option Pricing

Running time: $O(kn^2)$ where there are $O(n^2)$ nodes and each node has $O(k)$ buckets.

To guarantee convergence, k needs to grow with n at least.