MAT1856/APM466 Assignment 2

Yanlin Li, Student #: 1003770305

March, 2021

Summarized answer

0.1 Problem

Given that gasoline prices are at $p_0 = 1$ dollar per litre today, and each week i the price can go up by 10% to $p_{i+1} = p_i \times 1.1$ with probability 50%, or down to $p_{i+1} = p_i/1.1$ also with probability 50%. The price protection option can be exercised for no more than 4 times a year.

The problems are:

1. Calculate the price of the price protection plan for both buyer (Mr. Hamilton, maximum 50 litres per week) and seller (Ms. Curie, maximum 50,000 litres per week) under the given gasoline price circumstances.
2. Decide the optimal exercise nodes in the option trees.

0.2 Answer

1. The prices are:
   (a) Price for Mr. Hamilton (call option, buyer): $53.01
   (b) Price for Ms. Curie (put option, seller): $53014.63
2. The optimal exercise nodes are:
   (a) For Mr. Hamilton (call option, buyer): Last four weeks when gasoline prices are higher than 1.
   (b) For Ms. Curie (put option, seller): Last four weeks when gasoline prices are lower than 1.

Algorithms

There are 3 steps towards our goal: Computing underlying price, computing 1-up/down swing and adding one additional up/down swing.

0.3 Computing underlying (gasoline) price

I designed a 105×53 matrix to store the data. Each column represents a certain week and each row represents a certain possible price. There are 53 weeks in one year, so there are 53 columns. The possible prices range from 1.1^{-52} to 1.1^{52}, so there are 105 rows.

In the underlying price matrix U, the entry $U[i, j]$ has value 1.1^{52-i} if this price can be reached in week j. Note that if the price is reachable at (i, j), $|52 - i|$ and j can only be both even or both odd. Otherwise the entry is zero. I looped through all entries and assigned a value to each entry using the criterion above. Then we get a matrix for underlying price.

The actual matrix is too large to visualize, so below (Figure 1) is an example of 5 week underlying matrix.

```
[[0.         0.         0.         0.         1.4641    ]
 [0.         0.         0.         1.331      0.        ]
 [0.         0.         1.21       0.         1.21      ]
 [0.         1.1        0.         1.1        0.        ]
 [1.         0.         1.         0.         1.        ]
 [0.         0.90909091 0.         0.90909091 0.        ]
 [0.         0.         0.82644628 0.         0.82644628]
 [0.         0.         0.         0.7513148  0.        ]
 [0.         0.         0.         0.         0.68301346]]
```

Figure 1: 5 week case of underlying matrix

0.4 Computing 1-up/down swing

This step can be separated into three parts: identifying probabilities, calculating possible prices for the last week and using backward propagation to get all the prices. First, I initiate two zero matrices of the same size as underlying matrix. In all loops, we examine the underlying matrix for whether an entry is a node in the decision tree. If it is a node, the corresponding position in underlying matrix should not be zero. The outputs for this step are the two matrices, one for 1-up/down swing and another is a matrix indicating whether a node is optimal to exercise or not (a node is 0 if not and 1 if optimal).

Some notations:
1. S: the underlying (gasoline) price
2. E: value if exercise
3. F: future expected value if not exercise
4. p: probability for underlying price to go up on each node
5. u: value in 1-up/down swing if underlying price go up
6. d: value in 1-up/down swing if underlying price go down

1. Identifying probabilities:

   For each node except those in the last columns, saying the node with underlying price of 1.1^k, the probability that the underlying price will go up is:

$$p = \frac{now - down}{up - down} = \frac{1.1^k - 1.1^{k-1}}{1.1^{k+1} - 1.1^{k-1}} = 0.476$$

2. Calculating possible prices for the last week:

   For each node,

   (a) Up swing / Call option: 1-up price = $max(0, S - 1)$

(b) Down swing / Put option: 1-down price = $max(0, 1 - S)$

3. Backward propagation for prices:

   Loop from the 52^{th} column to the first column, for each node:

   (a) Value if exercise on this node:

      i. Up swing / Call option: $E = S - 1$

      ii. Down swing / Put option: $E = 1 - S$

   (b) If not exercise on this node, the future expected value is: $F = pu + (1 - p)d$

   (c) Compare the value if exercise or not exercise. If $E > F$, then it is optimal to exercise, set the corresponding node in exercise matrix to be 1 and the 1-up/down price is E. Otherwise, the 1-up/down price is F. Or more clearly, 1-up/down price = $max(E, F)$

The final price now for 1-up/down swings are both 0.268.

Note 1: indifference: Here I will assume that if the payoff of exercising and not exercising are the same, we tend to not exercise. The indifferent nodes are not treated to be optimal nodes. The reason is that if not exercise, the remaining option will still act as a price insurance and will have time value. Besides, I can find a very good pattern for optimal nodes if indifferent nodes are not exercised.

Note 2: Rounding problem: Because of the rounding problem in Python, the values which are actually the same may appear to be different, causing problems when deciding whether to exercise. So I will "accept as equivalent" if there's less than a 0.01% difference between exercising and waiting until the next period.

0.5 Adding one additional up/down swing

Our goal is to calculate k+1-up/down swing prices given underlying prices, k-up/down prices, exercise matrix for k-up/down swing and probability for underlying price to go up.

Some notations:

1. S: the underlying (gasoline) price
2. E: value if exercise
3. F_{k+1}: future expected value for k+1-up/down swing if not exercise
4. F_k: corresponding node for k-up/down swing
5. p: probability for underlying price to go up on each node
6. u: value in 1-up/down swing if underlying price go up
7. d: value in 1-up/down swing if underlying price go down

We make a copy of the k-up/down exercise matrix and k-up/down prices. All the changes will be made to the copied matrices, so that the algorithm will not change the input matrices. Loop through each node, use backward propagation, from the last week to the first week:

1. Excess value if exercise on this node:

   (a) Up swing / Call option: $E = S - 1$

   (b) Down swing / Put option: $E = 1 - S$

2. If not exercise on this node, the future expected value is: $F_{k+1} = pu + (1 - p)d$

3. There are two cases:

(a) If $F_k + E > F_{k+1}$ and this node has not been exercised before (the corresponding node in exercise matrix is 0): Exercise on this node and this is an optimal exercise node. k+1-up/down price is $F_k + E$. Set the node in the new exercise matrix to be 1.

(b) If $F_k + E \leq F_{k+1}$ or this node has already been exercised before (the corresponding node in exercise matrix is 1): k+1-up/down price is $max(F_k, F_{k+1})$. Here we use a maximum to take into consideration previous exercising processes of both the weeks afterwards and on this node before.

The same two notes here as in the step before.

This algorithm can be written into a single function. To calculate 4-up/down price, this function was applied four times, with each time using the previous outputs. Below is a table (Table 1) indicating how prices change when we go from 1-up/down swing to 4-up/down swing.

	Up	Down
1	0.268	0.268
2	0.535	0.535
3	0.798	0.768
4	1.060	1.060

Table 1: Prices from 1-up/down swing to 4-up/down swing

Results

We know from the Algorithm part that the 4-up/down swing option prices for one litre of gasoline each time are both \$1.060. Given that Mr. Hamilton's car consumes 50 litres of gasoline each week and Ms. Curie's refinery produces 50,000 litres of gasoline every week, we can multiply these two values to our prices. So the final prices for them are:

1. Price for Mr. Hamilton (call option, buyer): $\$1.060 \times 50 = \$53.01$
2. Price for Ms. Curie (put option, seller): $\$1.060 \times 50,000 = \$53014.63$

We can use the output from the exercise matrix to determine which nodes are optimal to exercise. In order to identify a clear pattern of the optimal nodes, I plotted the two graphs on the next page (Figure 2, Figure 3). Black dots are all the nodes that are not optimal to exercise while red dots represent all the optimal nodes. Y-axis is the underlying (gasoline) price for the corresponding node and X-axis is which week the node is in.

From the plots, it is clear that for both options, a node is optimal to exercise if it is in the last four weeks and the payoff of exercising this node is positive. More precisely, the optimal exercise nodes are:

1. For Mr. Hamilton (call option, buyer): Last four weeks when gasoline prices are higher than 1.
2. For Ms. Curie (put option, seller): Last four weeks when gasoline prices are lower than 1.

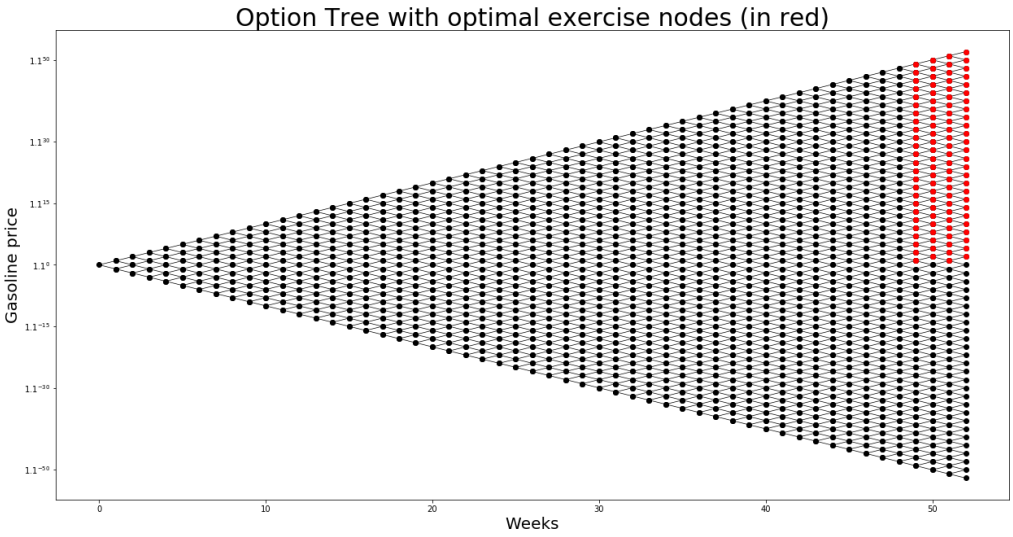

Figure 2: Call option optimal nodes

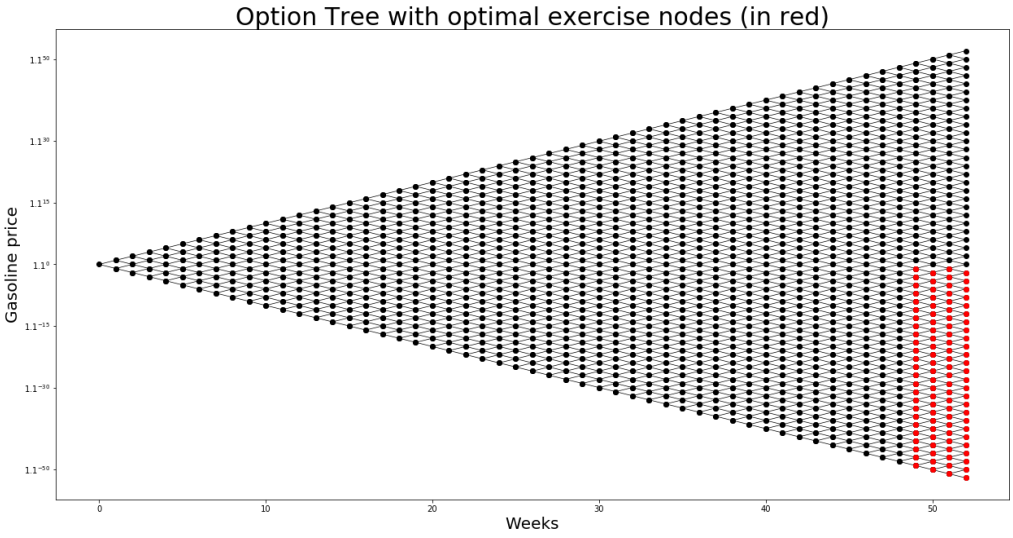

Figure 3: Put option optimal nodes

References

1. Fruit & Anton Menshov. (2019, July 15). Plot lattice tree in python. Retrieved March 10, 2021, from https://stackoverflow.com/questions/33712179/plot-lattice-tree-in-python