Quantitative Risk Management
Homework for Week 2

John Dodson

September 20, 2021

Solutions to these problems are due at the beginning of the next session, which is 5:30 PM on Monday, September 27. Please turn in your solutions to the TA. Include your U of Minn. student identification number on your submission to facilitate recording marks in the Canvas learning management system. Also include the names of any classmates you consulted with in developing your solutions.

Entropy

We will see below that options prices provide information about the distribution of the terminal value of the underlying interest. For each date $t < T$, where T is the expiration date of an option assume that $S_T|\mathcal{F}_t$ is a non-degenerate random variable with distribution function

$$x \mapsto F_t(x) = \mathrm{P}\left[S_T < x|\mathcal{F}_t\right] \quad \forall x > 0, t < T$$

of some measure of the corresponding events on the sigma-algebra $\mathcal{F}_t$ generated by the observed sample path[1] of the value of the underlying interest $\{(s, S_s) : s \leq t\}$.

Say we only have a discrete set of $\{(t, x, F_t(x))\}$. In particular, say we have

$$q_{i,j} = F_{t_j}(x_{i+1}) - F_{t_j}(x_i) = \mathrm{P}\left[x_i \leq S_T < x_{i+1}|\mathcal{F}_{t_j}\right] \tag{1}$$

for $j = 1, 2, \ldots, M$ and $i = 1, 2, \ldots, N$, x_i and t_j both increasing in their indexes. $q_{i,j}$ is the (forward) risk-neutral probability on date t_j that the terminal underlying value will be between x_i and x_{i+1}. Take $x_0 = 0$, $x_{N+1} = \infty$, and $t_{M+1} = T$. $\left(\{q_{i,j} : i \in \{0, 1, \ldots, N\}\}\right)_j$ is essentially a sequence of probability mass functions.

The risk-neutral **entropy** of the terminal value on date t_j is

$$H_j = \sum_{i=0}^{N} q_{i,j} \log \frac{1}{q_{i,j}} \tag{2}$$

where we take $F_{t_j}(x_0) = 0$ and $F_{t_j}(x_{N+1}) = 1$. This is the discrete analog of differential entropy, which for a general random variable X with distribution F is defined by $\mathrm{H}[X] = \mathrm{E}\left[-\log F'(X)\right]$. Note that since[2] $\lim_{q\downarrow 0} q \log(1/q) = 0$, we can omit terms for which $q_{i,j}$ is zero.

As time t advances towards T, we might expect that the entropy of $S_T|\mathcal{F}_t$ to decline to zero if S_t is a stochastic process with independent increments[3]. In this week's assignment, we will check this using options prices.

[1] The state space for stochastic processes contains sample paths; but European-style options are only sensitive to terminal values.

[2] Apply l'Hôpital's rule.

[3] $H_M = 0$ because almost all events for $S_T|\mathcal{F}_T$ have zero measure.

Risk-neutral probability mass function

Under the no-arbitrage assumption, the value of a non-defaultable European-style put contract on an investible underlying is the discounted forward risk-neutral expected value of the terminal value $\max(0, K - S_T)$ for strike price K and underlying price process $\{(t, S_t)(\omega) : 0 < t \leq T, \omega \in \mathbb{F}\}$ under the filtration $\mathbb{F}$ where $\omega_t \in \mathcal{F}_t \in \mathbb{F}$.

$$p_t(K) = d_t \, \mathrm{E}^{\mathbb{Q}^T} \left[\max(0, K - S_T) | \mathcal{F}_t\right]$$

where d_t is the discount factor (the risk-free present value at date t of a unit of account paid on date T).

Say you have $0 < K_1 < K_2$. Since the underlying and expiration date for both puts is the same, the value of the difference turns out to be[4]

$$p_t(K_2) - p_t(K_1) = d_t \int_{K_1}^{K_2} F_t(x) \, dx$$

We know from the mean value theorem that

$$F_t(K_1^\star) = \frac{p_t(K_2) - p_t(K_1)}{d_t(K_2 - K_1)} \tag{3}$$

for some $K_1^\star : K_1 < K_1^\star < K_2$. In particular, if $0 < K_1 < K_2 < K_3 < \ldots$ then we can assemble $K_1^\star < K_2^\star < \ldots$, which we can identify with $x_1, x_2, \ldots$ in the analysis above (even if we do not know their exact values).

Problems

I have assembled some daily closing prices for the (European-style) NASDAQ-100 index option expiring on September 17, 2021 in `https://www-users.cse.umn.edu/~dodso013/fm503/docs/fall2ex.html`.

N.B.: Market data can be inconsistent. In reality, the price of a commodity—even in a liquid market—depends on whether you are buying or selling, and on how much. Also, disseminating and processing the order book takes time; so quotes may not be exactly synchronous, even at the close of the trading session. We can attempt to determine a snapshot of prices for historical analysis; but sometimes the result is not satisfactory.

1. Our calculation depends on $p(K)$ being a non-decreasing function. Since the provided data do not strictly satisfy this requirement, explain how will you modify or filter the data to make it usable. **(2 points)**

2. Estimate the entropy of the terminal value of the index for the days in provided data. **(6 points)**

3. Discuss your results. Does it decline each day? **(2 points)**

Solutions

As I mentioned in the syllabus, this some of the assignments in this module requires familiarity with scientific computing. Software that facilitates interactive scientific computing include MATLAB™, Mathematica™, R, Python, and Julia. Demonstrations and solutions that I present will use Julia `https://julialang.org/`. Note that Julia syntax is similar to that of MATLAB™, with prepended dots to represent vectorized binary operations and array indexes starting at one.

[4] Apply integration by parts.

Data Filtering

The data we have approximates the theoretical premium curve $p(K)$. No-arbitrage theory requires that $p''(K) > 0$, i.e. the premium be convex in strike. Ideally, we would identify a maximal subset (indexed by $i = 1, \ldots, N$) per trade date such that

$$\frac{p_i - p_{i-1}}{K_i - K_{i-1}} > \frac{p_{i-1} - p_{i-2}}{K_{i-1} - K_{i-2}}$$

for $i = 2, \ldots, N$ strictly (with $p_0 = K_0 = 0$). This is a computationally hard problem, out of scope for this assignment. I expect that you chose your set manually; but I want to introduce a (albeit sub-optimal) automated approach here.

A necessary (but insufficient) condition for the convex subsequence is the property that p_i/K_i be strictly increasing. Identifying a longest increasing subsequence ("LIS") is a familiar problem in computer science; and a number of algorithms are available. One reasonable approach is to identify a longest common subsequence ("LCS") versus a sorted version. If we are looking for a *strictly* increasing subsequence, we can sort without duplicates, which I achieve below using the `unique` function.

```
"longest strictly increasing subsequence index array"
lisperm = v -> lcsperm(v,sort(unique(v)))
```

The version of the LCS algorithm that I used is listed in the appendix.

Calculation

My implementation of the entropy calculation is below. It takes in a discount factor, an array of strike prices, and a parallel array of put premiums. You can see in `idx₁` and `idx₂` the filtering I applied. But since this is not sufficient to guarantee positive probabilities, I also applies LIS to the derived empirical distribution function per `idx₃`.

```
"entropy from put prices"
function H(d::Real,K::Vector{Real},p::Vector{Real})::Real
    idx₁ = p.>0. # filter out zero premiums
    p₁ = p[idx₁]
    K₁ = K[idx₁]
    idx₂ = lisperm(p₁./K₁) # filter out non-increasing premiums
    p₂ = [0.;p₁[idx₂]]
    K₂ = [0.;K₁[idx₂]] # add zero-strike put
    F = diff(p₂/d)./diff(K₂)
    F₂ = [0.;F[F.<1.];1.] # filter out ITM slopes higher than the discount
    idx₃ = lisperm(F₂) # filter out non-convex premiums
    F₃ = F₂[idx₃]
    q = diff(F₃)
    return sum(q.*-log.(q))
end
```

Note that I affix zero/one to the beginning/end of the sequence of cumulative probabilities in `F₂`. Also note that this implementation assumes that the rows are sorted in ascending order of strike price and that there are no duplicates. In an automated setting, you should code more robustly with regards to these assumptions (e.g. sort and check for duplicates).

trade date	entropy
2021-09-02	4.626
2021-09-03	4.718
2021-09-07	4.488
2021-09-08	4.485
2021-09-09	3.972
2021-09-10	4.054
2021-09-13	4.374
2021-09-14	3.866
2021-09-15	3.826
2021-09-16	3.179

Table 1: Risk-neutral entropies for the September 17, 2021, value of the NASDAQ-100 index based on listed puts.

Results

Results are in table 1. Our estimate of the entropy of the September 17$^{\text{th}}$ opening value of the NASDAQ-100 index seem to generally decline most days in the weeks before, but not uniformly; and there is a notable increase on Friday, September 10$^{\text{th}}$ and over the following weekend. Market commentary at the time pointed to the slowing economic recovery, inflation fears, and stricter regulation of Chinese tech and finance companies.

Also, notice that the entropy was still sizable at the close of the session on September 17$^{\text{th}}$. This is surprising, since the option settles to the *opening* price of the index the following morning (at which point the entropy collapses to zero). So it seems there is more entropy priced into the last overnight than there is in the prior two weeks.

Discussion

There is a general result about the entropy of the sum of independent random variables, say X, Y:

$$\max\left(\mathrm{H}[X], \mathrm{H}[Y]\right) \leq \mathrm{H}[X+Y] \leq \mathrm{H}[X] + \mathrm{H}[Y]$$

The upper bound is relevant here. Consider S_T as a composition of three quantities: the value of the index at the close of September 9$^{\text{th}}$, S_0; the (log) return, X, between the close of the 9$^{\text{th}}$ and the close of the 13$^{\text{th}}$; and the (log) return, Y, between the close of the 13$^{\text{th}}$ and the open of the 17$^{\text{th}}$ when the options expire. We have:

$$S_T = S_0 e^X e^Y$$

This is not quite a sum of independent random variables, but it would be if we were modeling with $\log S_T$ instead of S_T.

$$\log S_T = \log S_0 + X + Y$$

Luckily, in the discrete version that does not matter! This is because logarithm is an increasing transformation so (1) is unchanged if we redefine the bins as $\log K_i^\star < \log S_T < \log K_{i+1}^\star$ rather than $K_i^\star < S_T < K_{i+1}^\star$.

So if the return between the close on the 13$^{\text{th}}$ and expiration were: (a) independent of the return between the 9$^{\text{th}}$ and the 13$^{\text{th}}$; and (b) identically distributed under the sigma-algebra for the 9$^{\text{th}}$ and the sigma-algebra for the 13th , then

$$\mathrm{H}\left[S_T|\mathcal{F}_{t+h}\right] \le \mathrm{H}\left[S_T|\mathcal{F}_t\right]$$

for $h > 0$ because $\mathrm{H}[\log S_t|\mathcal{F}_t] = 0$.

That is not what we see. The remaining entropy on the 13th is seemingly *higher* than it was on the 9th . So either assumptions (a) or (b) or both are not supported by this data.

Since (a) is tied to the efficient market hypothesis, this is hard to give up. On the other hand giving up (b) is not unreasonable; but it does present some econometric modeling challenges which we will have to face, starting with this week's material.

In this particular case, the narrative might be that options market makers on Friday, September 10th , revised higher their forecasts for the dispersion of the NASDAQ-100 index for the following week, based on the economic and political news and the market's reaction to that news.

Finally, let's discuss the last day. Why might the entropy still be so high after the close of the last trading session before expiration? This might be a forecast of the uncertainty associated with the overnight and the processing for determining the opening prices of the index constituents the following morning. It also might be related to the difficulty market-makers face in re-hedging in the overnight market.

Appendix: Longest common subsequence

This returns a vector of the indexes of `A` that make up a longest common subsequence with `B`.

```julia
"longest common subsequence index array"
function lcsperm(A::Vector,B::Vector)::Vector{Int}
    lens = zeros(Int,length(A)+1,length(B)+1)
    for (i,x) in enumerate(A), (j,y) in enumerate(B)
        if x == y
            lens[i+1,j+1] = lens[i,j]+1
        else
            lens[i+1,j+1] = max(lens[i+1,j],lens[i,j+1])
        end
    end
    idx = []
    x,y = size(lens)
    while x > 1 && y > 1
        if lens[x,y] == lens[x-1,y]
            x -= 1
        elseif lens[x,y] == lens[x,y-1]
            y -= 1
        else
            idx = [x-1;idx]
            x -= 1
            y -= 1
        end
    end
    return idx
end
```