High Frequency Market Making in the Foreign Market

Shahram Nikbakhtian

September 7, 2016

Table of contents I

Table of contents II

What is Market Making?

A market makes is a firm that stands ready every second of the trading day with a firm as and bid price. This is good for you, because when you place a market order to sell your thousands shares of Disney, the market maker actually purchase the stock from you, even if he doesn't have a seller lined up. In doing so, they are literally "making a market" for the stock. Without market makers, it would take considerably longer for buyers and sellers to be matched up with one another, reducing liquidity and potentially increasing trading costs as it became more difficult to enter or exit positions.

How are they remunerated?

They must be compensated for the service that they provide and the the risk that they take. What would happen if he buys your shares of Google and the stock price starts to fall before somebody wants to buy from him. In practice it is even worse as once the price starts to fall, the potential buyer would wait that market stabilize before he makes any decision. In order to avoid this he will decide to buy at 100 dollar and sell at 100.5 dollar. The difference here is only 0.05 dollar. And if he is an expert to attract more clients in the day he can manage to generate a reasonable amount of money and offset his risk.

Why is he called Market Maker?

So far it looks like he is a broker who manages to match buyer and seller. If it was so easy to find both sides(buyer and seller) so quickly and at the same time we woulkdd not call him market maker as he does not make the market. This is brokerage. In reality all market maker dreams this situation and make an effort that this scenario happens as many as possible. But it is not essential.

What is making the market?

Now he has to make the market. For example he has a website that only his clients have access (you might a sky why only his clients?). There he can type two numbers one for buying and the second for selling. To do it efficiently he has to know about the customers's demand and the supply of the Google' shares. And of course his own position would be reflected there, also he should be aware of the market other competitors are making. And if he is really very good in his job, he might know what is likely to happen, I.e the economy , news announcements etc.

What is making the market?

In very simple words a MM makes money just by trying to buy cheaper and sell more expensive. This is a bit tricky and is different from making money on fees or spread. A very important point here is that a MM has always the obligation to be ready to buy and sell. But at the same time he should if not he will lose his reputation as a good MM. And reputation here is the main and most important core of the business.

Market making is risky

Oppositely to what people believe market making is risky. A client wants to buy, the MM sell at 100 dollar as he agreed. And there is no client who wants to buy. What can he do? He waits and during this time, the price goes up and it is 101 dollar. Should he wait more? Or buy at 101 dollar. He waits more and the price goes higher and is 102 dollar now. It is enough, he buys at 102dollar and he lose 2 dollar. This scenario happens in practice every day and many time during each day. But if you are a successful MM the profitable scenario happens more than losing scenarios.

Market making is risky

At any time a MM announce that he is ready to buy and sell some amount of security at a specific price. For example buy/sell one share of yahoo for 99dollar/100dollar. The difference between them 100-99 = 1 is called the spread. Mid price is the average between them and is equal to 99.5. The buy price 99 is called the bid price and the sell price 100 is also called the ask price. Let's assume there are two MM and they announce their request to an exchange. And let's assume the second MM announce 99.2/100.2 as bid/ask. If you like to buy you prefer MM1 because he sells at the cheaper price 100 but if you want to sell you prefer MM2 because he buys at the higher price 99.2. The best price available in the market is the highest bid and lowest ask price. In this example is 99.2/100.

Some terminology

1. Order : is an instruction to buy or sell on a trading venue. These orders can have some complicated forms.
2. Market Order : is a buy or sell order to be executed immediately at current market prices. It is used when certainty of execution is a priority over price of execution.
3. Limit Order : It is an order to buy at no more than a specific price or to sell at no less than a specific price (called better price). It is used when one wished to control price rather than certainty of execution.
4. Order Book : is an academic list of buy and sell orders for a specific security.

Some terminology

1. Pips: relates to the smallest price movement any exchange rate can make. MOst currencies are quotes to 4 decimal places. And the smallest change would be in the last digit. This means one pips equals to 1/100th of a percent., or one basis point. SO if a currency pair move from 1.2234 to 1.2236, this would be a change of 2 pips. So if for example you have 10000 dollar in your account for every 1 pip move that EUR/USD makes in the market you will gain or lose 1 dollar.

Some HFT strategies

A number of different strategies are pursued by HFT firms in the FX market. The unifying characteristic is the method of implementing the trading strategies using sophisticated quantitative models and high speed. The various strategies can be classified as follows:

Some HFT strategies

1. Classic arbitrage exploits the differences between market prices and prices implied by no arbitrage conditions. If the price gaps are large enough to cover transaction costs, trades can be executed to lock in a risk-free profit. One typical arbitrage used to be done across the spot and futures prices of the same currency pair.

Some HFT strategies

2. Latency arbitrage exploits the small time lag between when market-moving trades take place and when market-makers update the prices they quote. By directly detecting potential price moves, the HFT player can profit from what it has learned ahead of other participants that rely on market-makers quotes.
3. Complex event processing includes a number of different strategies. They aim at detecting profit opportunities by exploiting various properties of currency prices such as momentum, mean-reversion, correlation (with other currency pairs or with other assets) and response to economic data releases.

Why is it Big data?

1. EBS updates every 20msec or 50 times per second.
2. 86400 seconds in one day.
3. FX market is a 24h/24h market.
4. There are 10 levels of price for each side (20 price)
5. 250 days in each day
6. One year data $= 50 \times 86400 \times 20 \times 8 \times 250 = 170\,Gig$
7. One day = 700 Meg
8. Hang on a minute, we have also the volume for each price. And one time-stamp for each row of data.

Model

Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards. We can formalise the RL problem as follows. The environment is a modelled as a stochastic finite state machine with inputs (actions sent from the agent) and outputs (observations and rewards sent to the agent)

1. a set of environment states S
2. a set of actions A
3. Probabilities of transitioning between states $T = P(S(t)|S(t-1), A(t))$
4. Observation (output) function $P(Y(t)|S(t), A(t))$
5. Rules that determine the scalar immediate reward of a transition $\mathrm{R} = E(r(t)|X(t), A(t))$

1. State transition function: $S(t) = f(S(t-1), Y(t), R(t), A(t))$
2. Policy/output function: $A(t) = \pi(S(t)))$

The agent's goal is to find a policy and state-update function so as to maximize the the expected sum of discounted rewards $E[R_0 + \gamma R_1 + \gamma^2 R_2 + ...] = E \sum_{t=0}^{\infty} \gamma^t R_t$

Model Sate

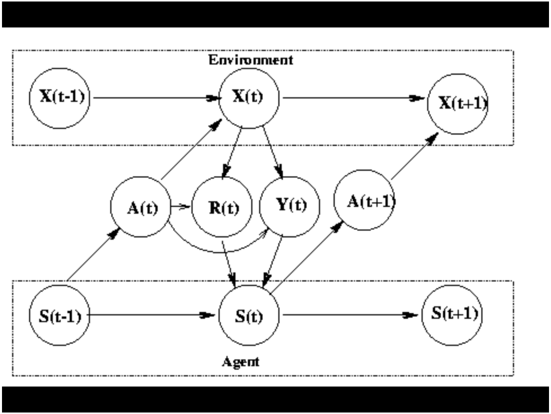

Figure: Model

In the special case that $Y(t) = X(t)$, we say that the world is fully observable, and the model becomes a Markov Decision Process (MDP). In this case, the agent does not need any internal state (memory) to act optimally.

MDP

A Markov Decision Process (MDP) is just like a Markov Chain, except the transition matrix depends on the action taken by the decision maker (agent) at each time step. The agent receives a reward, which depends on the action and the state. The goal is to find a function, called a policy, which specifies which action to take in each state, so as to maximize some function (e.g., the mean or expected discounted sum) of the sequence of rewards.

MDP

- $T(s, a, s') = Pr[S(t+1) = s' | S(t) = s, A(t) = a]$
- $R(s, a, s') = E[R(t+1) | S(t) = a, A(t) = a, S(t+1) = s']$

Value Iteration

Let us assume that we have a function V that associates to each state s a lower bound on the optimal total reward V^*. For example it can be $V_0 = 0$. At each state we can choose the action that maximize the expected reward of the present action + estimate total reward from the next step onwards. Using this idea we can get an update V_{i+1} on the V_i.

Value Iteration

In mathematical terms

- $V_0(s) = 0$ for all s.
- iterate for all s.

$$V_{i+1}(s) = \max_a E[R(s,a,s') + \gamma V_i(s')]$$
$$= \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_i(s')]$$

until $|V_{i+1} - V_i| < \epsilon$

Policy Iteration

Given any policy π_0 with corresponding V_{π_0} we can always get another policy π_1 that is at least as good. For each state s , choose the action that maximizes the expected total reward that will be collected if the previous policy is used from the next step onwards.

$$\begin{aligned}\pi_1(s) &= \arg\max_a E[R(s,a,s') + \gamma V(s')] \\ &= \arg\max_a \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_{\pi_0}(s')]\end{aligned}$$

Policy Iteration

Policy Iteration Algorithm

- Start with an arbitrary policy π
- iterate:
  1. Policy evaluation: solve the linear system

$$V(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V(s')]$$

  2. Policy improvement: for each $s \in S$

$$\pi(s) = \arg\max_{a} \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V(s')]$$

Reinforcement Learning

What happens if we dont have the whole MDP?

- We know the states and actions
- We dont have the system model (transition function) or reward function

Were only allowed to sample from the MDP

- Can observe experiences (s, a, r, s)
- Need to perform actions to generate new experiences

A Reinforcement Learning Market Making I

Environment Sates:

- Position of the Market Maker
- Order imbalance
- Bid-Ask spread
- Time-to-fill a limit order
- Price volatility
- Trading volume

A Reinforcement Learning Market Making I

Market maker action:

- Change the bid price
- Change the ask price
- Set the bid size
- Set the ask size
- Send the market order

Reward:

- Profit
- Reducing the Risk

A Reinforcement Learning Market Making II

Environment Sates:

- The ask and bid orders represent as a $2n$ vector and the ask/bid volumes is represented positives/negatives

Market maker action:

- Limit buy order
- Limit sell order
- Market buy order
- Market sell order

Reward:

- Profit

OLS

Let X be an $n \times k$ matrix where we have observations on k independent variables for n observations. Our statistical model will essentially look something like the following:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ \vdots \\ Y_n \end{pmatrix}_{n\times 1} = \begin{pmatrix} 1 & X_{11} & X_{21} & \dots & X_{k1} \\ 1 & X_{12} & X_{22} & \dots & X_{k2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & X_{1n} & X_{2n} & \dots & X_{kn} \end{pmatrix}_{n\times k} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \vdots \\ \beta_n \end{pmatrix}_{k\times 1} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \vdots \\ \epsilon_n \end{pmatrix}_{n\times 1}$$

in matrix form : $Y = X\beta + \epsilon$

OLS

The goal of least-squares estimation is to minimize the sum of squared differences between the fitted and observed values.

$$L(\beta) = \sum_i (Y_i - \hat{Y}_i)^2 = \|Y - X\beta\|^2$$

To find the minimum we need the gradient of $L(\beta)$ with respect to β. Let $g(\beta) = Y - X\beta$ and $f(x) = \sum_i x_i^2$ using the multivariate chain rule one gets:

$$\frac{\partial L}{\partial \beta} = -2X^T(Y - X\beta) = 0$$

Equivalently $\beta = (X^T X)^{-1} X^T Y$.

Matrix Product

To compute the estimated coefficients we need to compute $X'X$. Let's review out linear algebra. Let's assume $A \in R^{m \times n}$ and $B \in R^{n \times p}$. Let's partition A and B by rows.

$$A = \begin{pmatrix} - & a_1 & - \\ - & a_2 & - \\ & \dots & \\ & \dots & \\ - & a_m & - \end{pmatrix} B = \begin{pmatrix} | & | & | & | \\ b_1 & b_2 & \dots & b_p \\ \vdots & \vdots & \vdots & \vdots \\ | & | & | & | \end{pmatrix}$$

Matrix Product

Then AB is a matrix of inner products

$$AB = \begin{pmatrix} a_1 b_1 & a_1 b_1 & \dots \\ a_2 b_1 & a_2 b_2 & \dots \\ \dots & \dots & \dots \end{pmatrix}$$

$$A = \begin{pmatrix} | & | & | & | \\ a_1 & a_2 & \dots & a_n \\ \vdots & \vdots & \vdots & \vdots \\ | & | & | & | \end{pmatrix} B = \begin{pmatrix} - & b_1 & - \\ - & b_2 & - \\ & \dots & \\ & \dots & \\ - & b_n & - \end{pmatrix}$$

AB can be represented as sum of outer products.

$AB = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$

Complexity

- $\beta = (X^T X)^{-1} X^T Y$.
- Computation Complexity: $O(nk^2 + k^3)$
- Storage: $O(nk + k^2)$ floats.

- Matrix multiply of $X^T X$: $O(nk^2)$ operations.
- Matrix inverse: $O(k^3)$ operations.
- $X^T X$ and its inverse: $O(k^2)$ floats
- X : $O(nk)$ floats

Using the QR decomposition, we can solve the least squares problem :

$$\min_{\beta} \|y - X\beta\|_2 \tag{1}$$

Where $X \in R^{m \times n}$, without forming the normal equations.(m number of samples and n number of regressors). To do this we use the fact that the Euclidean vector norm is invariant under orthogonal transformations Q

$$\|Qy\|_2 = \|y\|_2 \tag{2}$$

Introducing the QR decomposition of as , we get $\|r\|_2 = \|y - X\beta\|_2 = \left\| y - Q \begin{pmatrix} R \\ 0 \end{pmatrix} \beta \right\|_2 = \left\| QQ^T y - Q \begin{pmatrix} R \\ 0 \end{pmatrix} \beta \right\|_2 = \left\| Q(Q^T y - \begin{pmatrix} R \\ 0 \end{pmatrix} \beta) \right\|_2 = \left\| Q^T y - \begin{pmatrix} R \\ 0 \end{pmatrix} \beta \right\|_2$.

Then we partition $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ where $Q_1 \in R^{m\times n}$ and denote:
$Q^T y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} Q_1^T y \\ Q_2^T y \end{pmatrix}$ Now we can write:
$\|r\|_2^2 = \left\| \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} R\beta \\ 0 \end{pmatrix} \right\|_2^2 = \|y_1 - R\beta\|_2^2 + \|y_2\|_2^2$ So to find β we have to solve $y_1 = R\beta$ therefore $\beta = R^{-1}Q_1^T y$.
So that was the theory. Now algorithmically we never inverse the instead we solve the equation by the back substitution technique. We dont need to store Q we only need R and y.
Ultimately the most important thing is how to update a new row or observation, which is important for big linear regression and online regression.

Actually we dont need to re-compute everything.

$$\begin{pmatrix} X & y \end{pmatrix} \rightarrow Q^T \begin{pmatrix} X & y \end{pmatrix} = \begin{pmatrix} R & Q_1^T y \\ 0 & Q_2^T y \end{pmatrix} \tag{3}$$

Now let the new observation to be x_{new} and y_{new}

$$\begin{pmatrix} X & y \\ x_{new} & y_{new} \end{pmatrix} \rightarrow \begin{pmatrix} Q^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X & y \\ x_{new} & y_{new} \end{pmatrix} = \begin{pmatrix} R & Q_1^T y \\ 0 & Q_2^T y \\ x_{new} & y_{new} \end{pmatrix} \tag{4}$$

It can be seen that the middle part of above (I mean $Q_2^T y$) doesnt play a role in the new R.

So remove it and we got this formulation:

$$\begin{pmatrix} R & y_1 \\ x_{new} & y_{new} \end{pmatrix} = \begin{pmatrix} \times & \dots & \times \\ \vdots & \ddots & \vdots \\ \times & \dots & \times \end{pmatrix} \tag{5}$$

The problem is that this new matrix is not upper triangular, but it is easy to transform it to an upper triangular matrix. The new matrix has only the last row which is non-zero. Doing the transformation we got the new