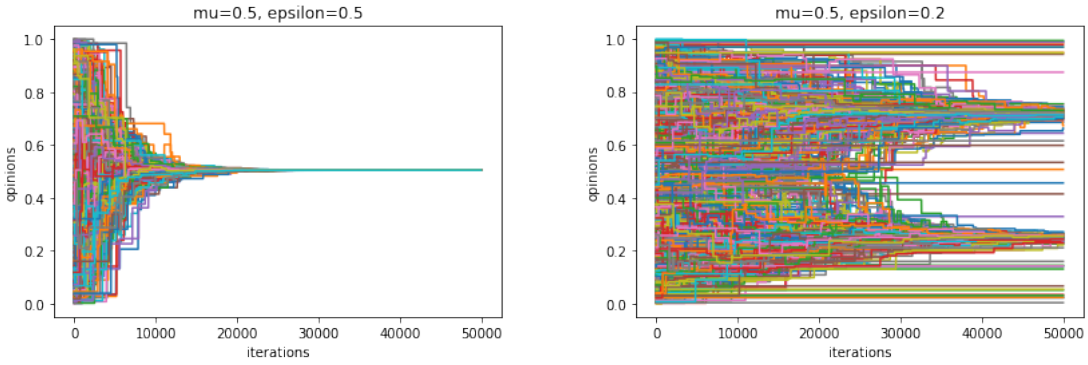
2.1 Opnion dynamics

2.1.1

I build the Barabasi Albert network for 1000 agents with 4 edges to attach from a new node to existing nodes. At each time step, one random agent i is chosen from the network to interact with one of its neighbours j. If the condition $|o_i$ - $o_j| < \epsilon$ is satisfied, they update their opinions. Otherwise they keep their opinions and go to next iteration.

2.1.2

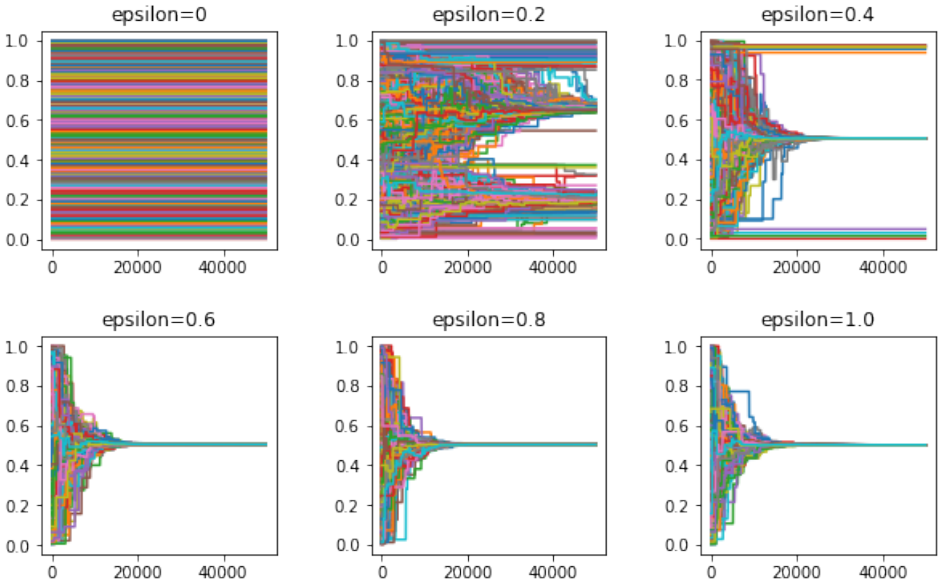
For the two given μ and ϵ pairs, I run the simulation 50000 times and the time charts of the opinions are as follows:



For $\epsilon = 0.5$ and $\mu = 0.5$, we see that opinions of each agent converge to 0.5 after around 20000 iterations. However, for $\epsilon = 0.2$ and $\mu = 0.5$, opinions are much more disorganized. Some of them tend to cluster at around 0.7, some of them tend to cluster at around 0.2, while others remain unchanged for the whole iterations. The reason for this will be discussed in the next subsection.

2.1.3

To see more clearly how the change of ϵ affects the opinion dynamic simulation, I set a step of 0.2 for ϵ in the range of 0 and 1 while μ remains a constant value 0.5. The following plots are drawn to make comparison:



Except for the two extrem case $\epsilon = 0$ and $\epsilon = 1$, we see for $\epsilon = 0.2$ and 0.4, there are several clusters of opinions. But for $\epsilon = 0.6$ and 0.8, all the opinions converge to 0.5. We can conclude that

as ϵ becomes bigger, there will be less clusters and the convergence is faster. This is understandable: two agents i and j meet and update their opinions under the condition $|o_i$ - $o_j| < \epsilon$. As ϵ becomes bigger, it's easier for the two agents to update their opinions (in this case taking average). Agents with opinions in a small interval may reach agreement and form clusters at first. For small ϵ, the opinions of different clusters will no longer be updated because they don't satisfy the condition. But for larger ϵ, by taking average of the clusters several times, they will gradually converge to neutral opinion (0.5) in the end.

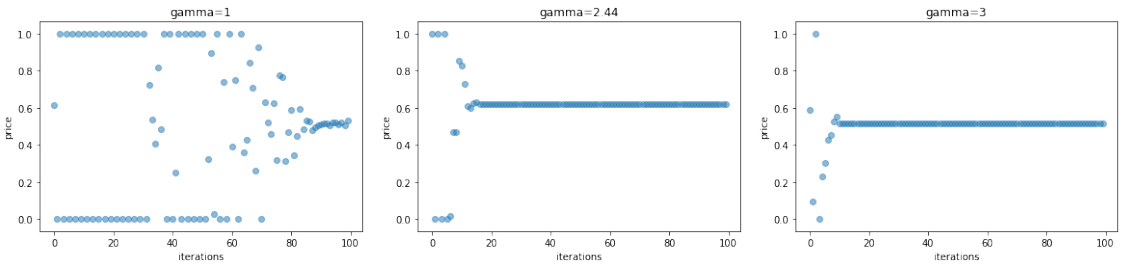
2.2 ABM for prediction markets

2.2.1

To build the prediction market model, we inherit deffuant model in 2.1 and take market price into consideration. In each time step, a certain fraction of agents are chosen to be active according to probability $P = (T - \tau)^{-\gamma}$. The market price will be updated based on the excess demand and then opinion dynamic process is performed as in 2.1. To make the simulation more realistic, I set the number of agents interacting with other agents in each time step to be 100. The other initializations are $\epsilon = 0.5$, $\mu = 0.5$, $m = 4$, duration $T = 100$ and initial price $\pi = 0.5$.

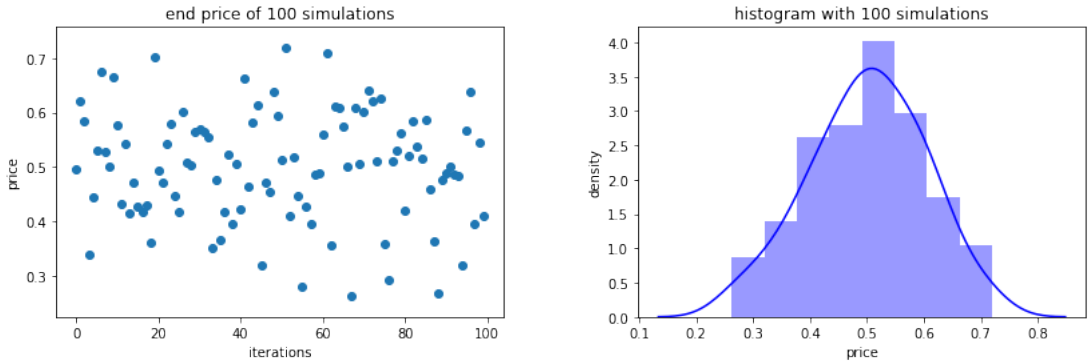
2.2.2

For the three scaling parameter γ=1, 2.44 and 3, the time plots for normalized price are as follows:



It shows that for $\gamma = 1$, the price fluctuates between 0 and 1 a lot and tend to converge to 0.5 in the end. But as γ increases, the price converges to 0.5 more quickly (at around 20 iterations for $\gamma = 2.44$ and at around 10 iterations for $\gamma = 3$). The mathematical interpretation is that the probability each agent participates in the market $P = (T - \tau)^{-\gamma}$ decreases exponentially as γ increases, i.e. nobody will participate in the market and the price will stay stable with small P.

2.2.3



If we run the simulation 100 times with $\gamma = 2.44$ and $T = 100$ and record the price at the end of each simulation, we can get the results above. The scatter plot shows that most of the end market prices fall in the range (0.4, 0.6), but there are also some outliners. From the histogram we can see

the price roughly follows normal distribution with peak at 0.5. As we see, each simulation of the same model may yield completely different results, so it's necessary to run a model several times and take average to reduce randomness. To get a same trend-line and reproduce the same exact results for a given set of parameters, we need to fix all the random choices in our model: the noise term v, the agents being active in each time step, and also the agents and their neighbours chosen to perform opinion dynamic.

2.3 Validation

2.3.1

The total number of unique markets Q = 3385;

The duration of the market for each market is stored in the dataframe named duration, the first 5 samples of them are:

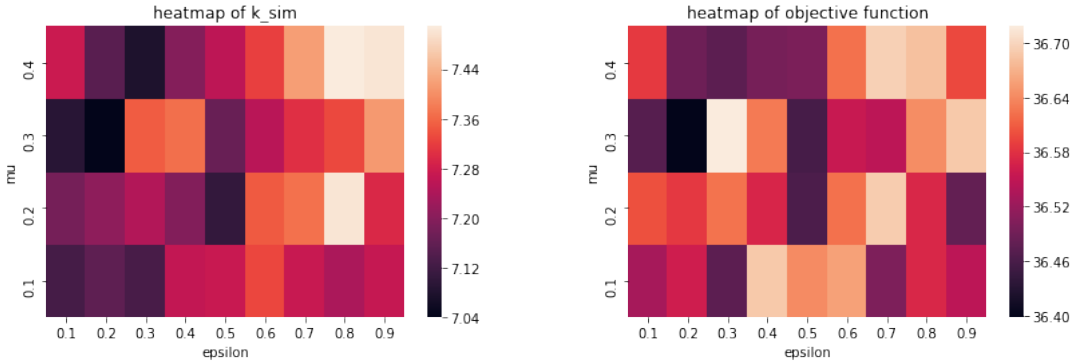
Table 1: market duration

market_id	433	434	435	436	437
duration	28	28	655	655	655

2.3.2

For ϵ in the interval (0, 1) and μ in the interval (0, 0.5) with a step size of 0.1, we extract each duration in the duration dataframe as T to run the simulation. And other initial parameters are the same as 2.2 with scaling parameter $\gamma = 2.44$. The whole dataset of return r_t and objective function of the returns f for each simulation are stored in **return.csv** and **obj_function.csv**.

For each pair of ϵ and μ, we get a kurtosis k_{sim} and an objective function f for each market and take their average to plot the heat map:



For the heat map of k_{sim}, we see that most of the kurtosis are in the range of (7.04, 7.44). For smaller ϵ the kurtosis seems to be smaller, but there is no such evidence for μ. The reason might be that as we see in 2.1.3, bigger ϵ results in faster convergence. As the price stays stable, there are a large bunch of zeros in the return list which causes high kurtosis.

For the objective functions $f = |k_{sim} - k_{emp}|$, we use the aggregate values $k_{emp} = 39.31$ for each simulation. As shown in the heat map, f are in the range of (36.4, 36.7). But there is no clear evidence of how ϵ and μ affect the scores.

The heat map of f shows that the current objective function may not be a very good validation of k_{sim} and k_{emp}. Ideally k_{sim} and k_{emp} should be the same and f will equal to zero in such case. There may be two reasons: we use the same k_{emp} to calculate f, which might vary a lot in simulations of different duration time T. Also the more zeros the return list has, the higher kurtosis it will get, resulting in higher f which is inaccurate. So to perform a better validation, we may use the CloseSharePrice for each market to get a more accurate k_{emp} for each simulation. Also we can add coefficient α for the objective function: f = α * $|k_{sim} - k_{emp}|$. We can assign smaller α for long market duration T and larger α for short market duration T to balance the lists of zeros.

3 Cryptocurrency market

3.1

To build an Agent-based model for the Bitcoin market, I referred to [1] and what we've learned in class. There are two kinds of agents interacting with each other: random traders and chartists. Random traders buy or sell with the same probability in spite of current price. Chartists issue buy orders when the price is increasing and sell orders when the price is decreasing, aiming to gain by placing orders. They issue orders when the price relative variation in a given time window is higher than a given threshold. The choice of the two kinds of agents as regard of the trading strategy is to simulate the fluctuations and stable periods of the real market. In short, chartists cause fluctuations in market and random traders keep the market price relatively stable in the long run.

3.2

In the Bitcoin system, the sorted lists of buy and sell orders are kept in an order book. Buy orders are sorted in descending order and sell orders are sorted in ascending order with respect to the price. And the imbalance between demand and supply causes price fluctuation. The interactions of the model are as follows:

Choose agents: At each simulation step t, a fraction of random traders and chartists are chosen to be active by probability P_r and P_c, and they choose to buy or sell based on their nature and current price $p(t)$. The amount of each buy or sell order depends on the fiat currency $c_i(t)$ or crypto-currecy $b_i(t)$ they hold.

Check availability: the first buy order b_i and the first sell order s_j of the lists are inspected to verify if they match. If $b_i > s_j$, a transaction occurs. The order with the smaller residual amount is fully executed, whereas the order with larger amount is only partially executed, and remains in the head of the list with residual amount reduced by the amount of the matching order.

Update the book and price: after the transaction, the next pair of orders at the head of the lists are checked for matching and so on until they do not match anymore. Then the book accept new orders generated in this time step and the price is updated:
- if $b_i > 0$ and $s_j > 0$, $p(t+1) = \frac{b_i + s_j}{2}$;
- otherwise if $b_i > 0$, $p(t+1) = min(b_i, p(t))$, if $s_j > 0$, $p(t+1) = max(s_j, p(t))$;
- if $b_i = s_j = 0$, $p(t+1) = p(t)$.

The interactions in this model are more complicated than what we learnt in class and can better simulate the real Bitcoin market. They take into consideration not only price but also amount of each transaction. Also not all the transactions can be satisfied in one time step, so a book of lists of orders is necessary to record the remaining buy and sell transactions for next step.

3.3

The **open datasets** can be found on BlockChain (https://blockchain.info/), which is a web site displaying detailed information about all transactions and Bitcoin blocks, providing graphs and statistics on different data. The empirical data (such as daily data about price, unique address number and bitcoin number) can be extracted from the website to verify the model[1].

References

[1] Luisanna Cocco, Giulio Concas, and Michele Marchesi. Using an artificial financial market for studying a cryptocurrency market. *Journal of Economic Interaction and Coordination*, 12(2):345–365, 2017.