Abstract

In this project, we evaluate the platform-wise effect on boosting the production of a social nudge sent from the follower to the creator. Given a self-generated following relationship network on the platform, we have conducted a precise calculation of the global effect. Also, an approximation and a downsample approximation are developed to avoid massive calculation in practical application.

Significance

As data science students immersed in the very environment with developed technology and fragmented entertainment, We can keenly feel the impact of social media on people's lives, as exemplified by short video platforms. Also as users of social media platforms, we have noticed that sending social nudges could significantly stimulate the creator's productivity.

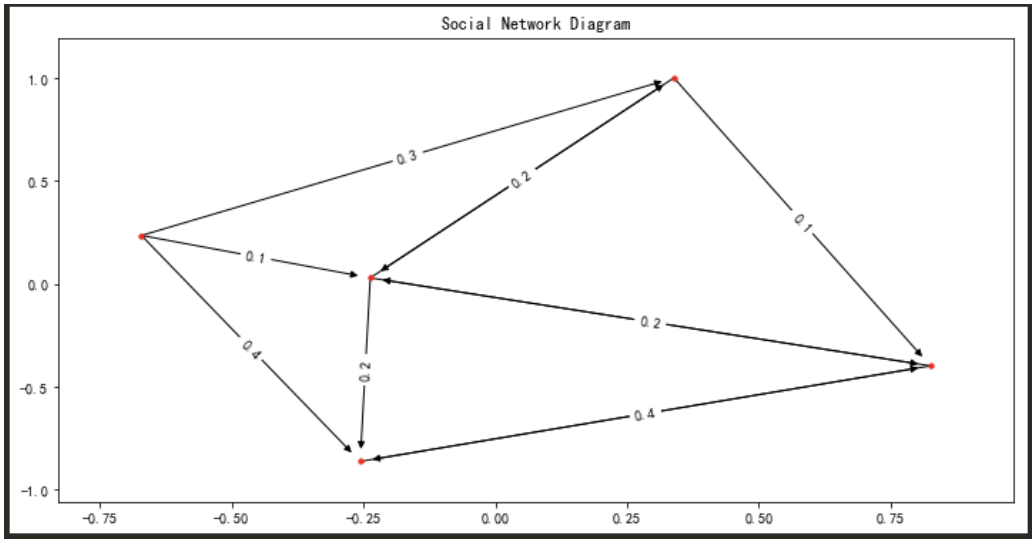
In this project, we quantify the global effect of sending one social nudge. This gives out a specific view of how users' activity could create value for the overall platform. With macro considerations, we believe this can provide a solid quantitative basis for the platform's business analysis.

Content

- Generating network

Given the probability of following *prob_f* (0 < prob_f < 1) and nodes number *node* (integer >= 2), we simulate a simple social platform network model with the following relationship only. Also, the user can choose to compliment the network generating by manually inputting a list *relation_list* representing the following relationship.

Weight *p_e* is assigned to each edge representing the peer-to-peer effect of sending a social nudge. Note here only followers can send nudges, and the friend relationship (two users both following each other) will increase the weight of sending social nudges.



Visualization of simulated social platform network with node = 5

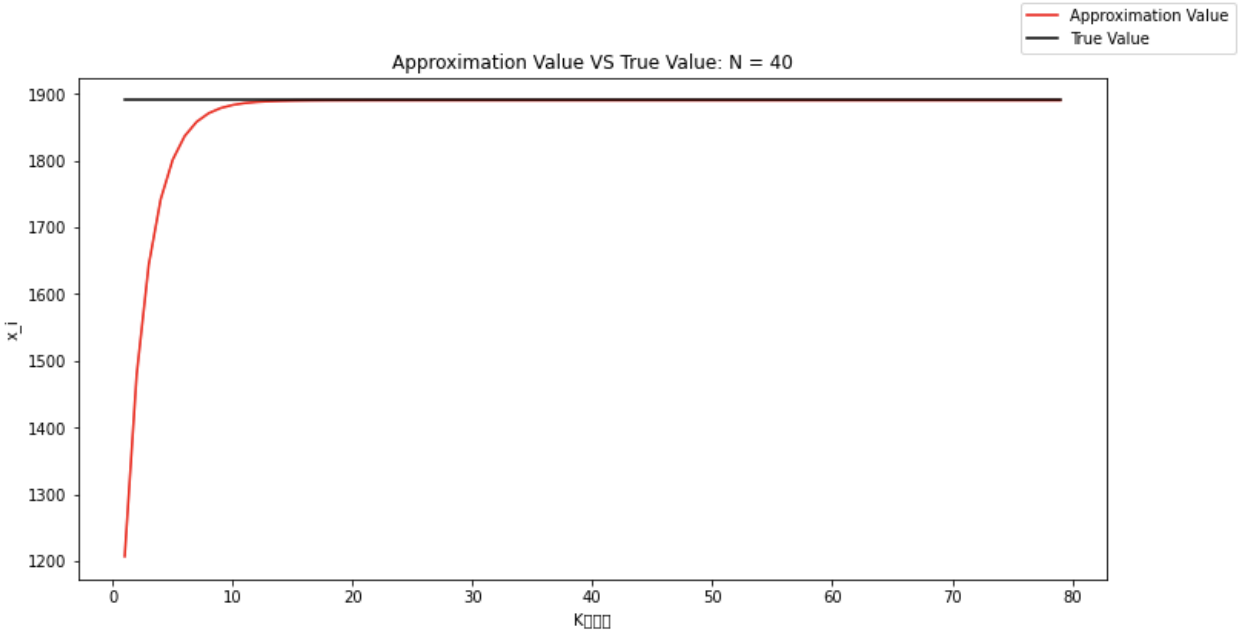
- Calculating true value

As the mathematical proof shown in the paper [link], the effect of social nudges on the whole platform-wise network is captured by the base nudging rate *mu_e* and its diffusion effect. Here we define the nudging rate *mu_e* (integer) as the intrinsic amount of nudge on edge *e*, and the intensity of social nudge diffusion, denoted by *dle*, also alpha_p (0 < alpha_p <1) denotes the time discounting factor of the direct production boosting effect, and alpha_d (0 < alpha_d <1) denotes the time discounting factor of nudge diffusion. Given all the parameters mentioned above, a matrix D capturing the first-order diffusion $D = (d_{le}: (l, e) \in V^2)$ can be presented. Under the restriction of $\left|\left|1/(1-\alpha_p)D\right|\right|_q \leq \delta$ for $\delta \in (0,1)$to make sure that $I - 1/(1-\alpha_d)D$ is invertible, we can calculate the true value of the social nudge effect.

To have the D satisfying the restriction, we would shrink D until it fulfills the requirement. Here the multiplier *shrink_size* helps to rescale D.

- Approximation

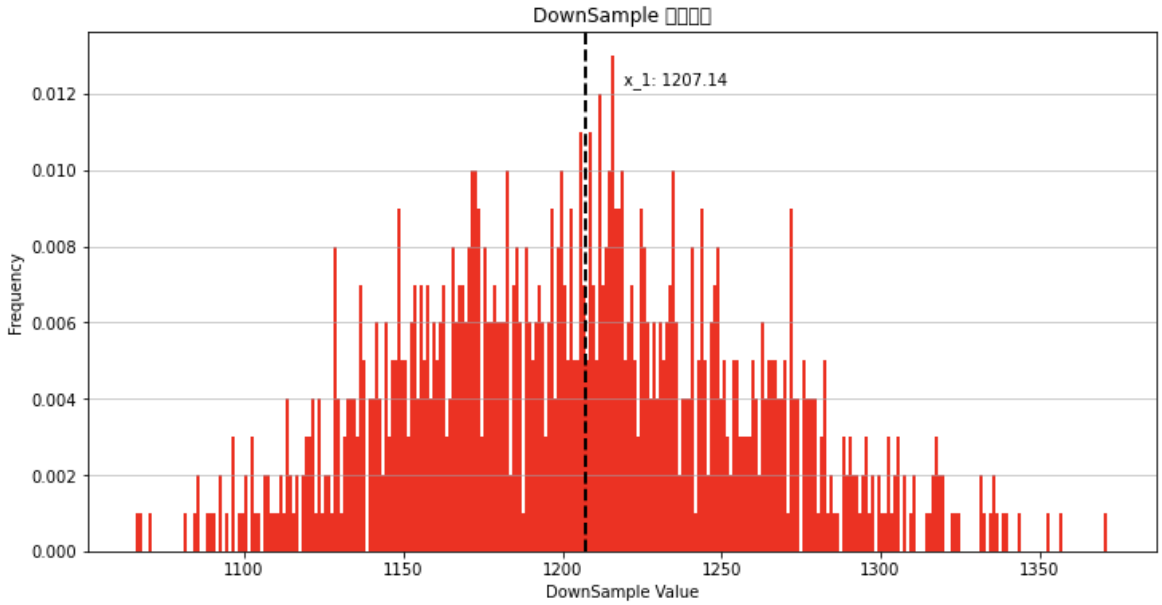
This is a way to approximate the social nudge effect eschewing the massive calculation of inversing matrix. Instead of calculating the ultimate effect of social nudge in one step of matrix inverse, the approximation is conducted by summing up the effect of each spread. As k goes to infinite, the approximation value should converge to the true value.



- Downsample approximation

Downsample approximation is an intuitive way to decrease complexity thus reduce the arithmetic difficulty. We conduct subsampling by randomly choosing several social nudge edges from the whole network, and rescale the result back to the full population. Significant variance and error can be expected from this method, but it would eventually have a distribution close to the true value with the increasing times of downsampling approximation.

In this approximation, the downsample size and downsample times should be determined according to the requirement of runtime, complexity, and accuracy.



Downsample result, with nodes = 40, sample size = 10, times = 1000. It is clear that the results centralize around the true value (dashed line).

Parameter explanation

- node: the number of nodes (users) in the network, positive integer;
- prob_f: the probability of following, 0 < prob_f < 1;
- p_e: social nudge effect (weight) on each edge;
- mu_e: basic nudging rate, i.e. the intrinsic amount of nudge on edge e, integer;
- alpha_p: the time discounting factor of the direct production boosting effect, 0 < alpha_p <1;
- alpha_d: the time discounting factor of nudge diffusion, 0 < alpha_p <1;
- d_l_e: intensity of social nudge diffusion, i.e., the expected increase of the number of nudges sent on edge e due to one nudge provider e receives on edge l directing to e;
- D: matrix capturing the first-order diffusion on all edge pairs, $D = (d_{le}: (l,e) \in V^2)$; satisfies $\left|\left|1/(1-\alpha_p)D\right|\right|_q \leq \delta$ for $\delta \in (0,1)$;
- shrink_size: a multiplier to D that keeps D under the regulation $\left|\left|1/(1-\alpha_p)D\right|\right|_q \leq \delta$ for $\delta \in (0,1)$, 0 < shrink_size < 1
- sample_size: the downsample size, sample_size < node;
- times: the times of downsampling.

Main Function explanation

- get_random_E(), get_input_E(): generating a list of edges with the following relationship;
- random_E_V_Network(), input_E_V_Network(): generating a platform-wise network based on the edges;
- get_random_D(), get_input_D(): generating a matrix capturing the first-order diffusion on all edge pairs;

- get_random_DF(), get_input_DF(): return a dataframe based on the the matrix D;
- check_normD(): check whether $I - 1/(1 - \alpha_d)D$ is invertiable. Shrink D with multiplying shrink_size if it is not invertiable;
- get_XY_star(): calculate the true value of social nudge effect;
- DownSample_Approximation(): generate downsample approximation;
- get_XY_k(): generate approximation with increasing k.

Future Improvements

- The current social nudge models support both the random system and the self-input system. However, the self-input model is tightly constrained by the input types and information formats. Therefore, we should develop the module to adapt to various input types.
- According to our tests, the network visualization was too convoluted when the node number was larger than 10. However, in reality, social networks are usually so complicated with more than thouthands of nodes, which means that we should design better visualization models.