IDEA ADOPTION PREDICTION

2014 Data Mining
Final Project 3

Team 4 R02924053 吳駿獻 R02943091 王詠文 B00901002 賴映安

Idea Adoption Prediction

1 Problem Formulation

Given: two input files

graph.txt, file containing a user social network.

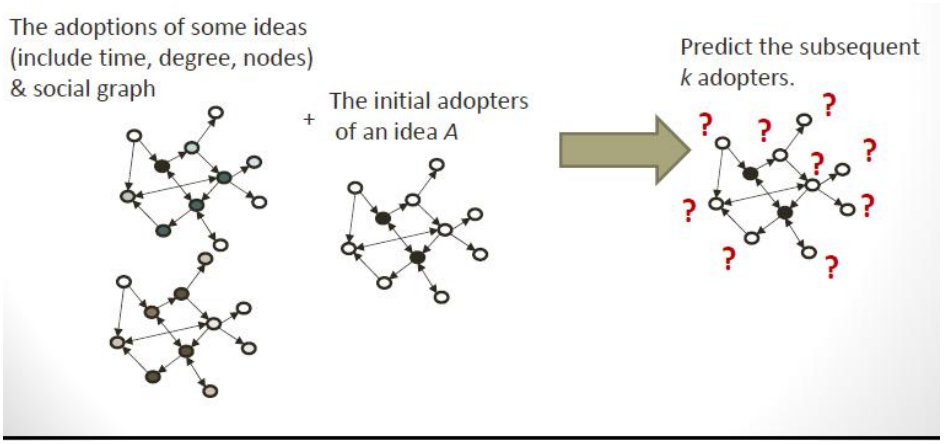
data format: node_id its_neighbor_1 its_neibhbor_2....

training.txt, file containing an idea adoptions list

data format: node_id idea_id time degree

Each line of the list records that a node adopts a certain idea in a specific time and the degree of this adoption is known.

Output: Given a test data containing a set of adopters of an idea, we want to find the subsequent 100 adopters of the idea.



Besides, our algorithm should beat the baseline stated in the slides of problem description. In this report, we will describe our method with care and discuss the reason behind some special design we did. Also the performance of each algorithm we proposed is visualized so as to differentiate the goodness of each algorithm.

2 Solutions

Matrix Factorization (MF)

Overview

Matrix factorization is to factorize a matrix, i.e. to find out two (or more) matrices such that when you multiply them you will get back the original matrix.

From an application point of view, matrix factorization can be used to discover latent features underlying the interactions between two different kinds of entities. And one obvious application is to predict ratings or degree of adoption in collaborative filtering.

Given that each user has some preferences to ideas in the system, we would like to predict whether the users would adopt the ideas that they have not yet adopted. In this case, all the information we have about the existing degrees can be represented in a matrix.

Assume now we have 5 users and 5 ideas, and degrees are numbers ranging from 0 to 1.0, the matrix may look something like this (a hyphen means that the degree is unknown):

	I1	*I2*	*I3*	*I4*	*I5*
U1	0.6	-	0.9	-	0.2
U2	0.8	-	0.3	-	0.3
U3	-	-	0.4	-	0.4
U4	0.2	1.0	0.2	0.5	-
U5	1.0	0.4	1.0	-	0.9

As a result, the task of predicting the missing degrees can be considered as filling in the blanks (the hyphens in the matrix) such that the values would be consistent with the existing degrees in the matrix.

The intuition behind using matrix factorization to solve this problem is that there should be some latent features that determine how a user adopts an idea. For example, two users would give high degrees to a certain post if they both like the topic of the post, or if the post is from certain group, which

is a genre preferred by both users. Hence, if we can discover these latent features, we should be able to predict the unknown values.

Mathematics of MF

We have a set U of users, and a set I of items. Let R of size $|U| \times |D|$ be the matrix that contains all the degrees that the users have assigned to the ideas. Also, we assume that we would like to discover K latent features. Then, our task is to find two matrices P(a $|U| \times K$ matrix) and Q (a $|D| \times K$ matrix) such that their product approximates R:

$$R \approx P \times Q^T$$

In this way, each row of P would represent the strength of the associations between a user and the features. Similarly, each row Q of would represent the strength of the associations between an idea and the features.
The way to find P and Q is first initialize the two matrices with some values, calculate how different their product is to R, and then try to minimize this difference iteratively (called gradient descent).

To minimize the error, we need to know which direction to modify the values and update the values.

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2$$

Here, α is a constant whose value determines the rate of approaching the minimum.
A common extension to this basic algorithm is to introduce regularization to avoid overfitting. This is done by adding a parameter **β** and modify the squared error as follows:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (|P|^2 + |Q|^2)$$

Mad Warrior Method

The Mad Warrior Method observes how many times that each user (node)

appears in training.txt file. In other words, we think that the frequent that a user adopts/dislike an idea, the more probability that he/she will adopts the test ideas. Therefore, we sort the users in the "training.txt" according to their number of presences. And then output the most 100 frequent idea-adopters from the list as the answers.

Simulated Propagation

Overview

In this method, we amount to simulate how ideas are spread over human social networks. We first evaluate the influence of each pair of friends relationship by applying evaluation function each time we observe that a users adopts/rejects an idea later than his/her friends. After that, we start to propagate each test idea. For each set of initial adopters given in testing files, we assign them initial scores as the their intendancies of adopting this testing idea. Then we propagate the idea by modified BFS search algorithm.

Evaluating Influences between Friends

We give a more formal definition of our influence evaluation here. Let u_a, u_b are pair of friends, and $\mathrm{n}(u)$ denote number of friends that user u have. Let I denote the set of ideas that u_a, u_b both adopts/rejects and u_a adopts earlier, and $\deg_a(i)$ denote the degree that user u_a adopts the idea i. Then the influence between u_a, u_b are defined as

$$\inf(u_a, u_b) = \sum_{i \in I} \deg_a(i) \times \deg_b(i) \times \exp\left(\frac{n(a)}{\Delta d_i}\right)$$

, where Δd_i is the difference of date that u_a and u_b adopts idea i in days.

First note that the influence is not symmetric, i.e. $\inf(u_a, u_b) \neq \inf(u_b, u_a)$. In plain words, how u_a affects u_b does not means how u_b affects u_a. We think that the more friends a user has, the more influence that he/she has. Also, the influence should be inversely related to how long the users. The longer time they adopt the same idea within each other, the less influence they have to each other.

Simulating Propagation

For each testing idea, we first give the initial adopters a score obtained from the libMF testing result. Then we perform breadth-first-search (BFS) graph search algorithm on the friendship network. However, we modified the algorithm so as to satisfy our need on simulating how ideas are propagated. Upon the searching agent visit a node, it adds the score of the last node the agents visit to that of this node (with each node's score initialed to testing results from libMF). Also, we consider that in real world, one may get influenced by many people rather than only one, therefore we increase the visit limit of each node to let our agent visit a node several times. The formula of how agent calculate the score when visiting a new node v is

$$score(v) = \inf(u, v) \times score(u) \times \exp(-steps)$$

, where that last exponential term is for influence attenuation, i.e. the influence from the initial adopter to the upfront node should decay according to the distance of them.

Finally, we sort the score and find out the most 100 highest score as our answer and beat the baseline.

Matrix Factorization with Mad Warrior Method

We combined Matrix Factorization and Mad Warrior Method. The principle is simple. We use Mad Warrior Method to pick first 200 candidates. And then use libMF to test the rate of these 200 candidates. Finally we output the first 100 candidates that have the highest rate obtained from Matrix Factorization method.

In particular, in this method we drop out the difference of the degrees. That is, we set all degree to the users in the idea adoption list to 1. The reason behind this is since we are combing Matrix Factorization with Mad Warrior Method (which only accounts frequencies of adoption), we don't need the degree anymore, just regard each adoption as "one event".

3 Results

We present the performances of the solutions we proposed in the following bar plot.