Data Mining Final Report (NTU, Fall 2014)

Yi-Hsiu Liao[1], Hsiang-Hung Lu[2], Sheng-Syun Shen[3]

Project 3: Adoption Prediction
[1]R03921048, [2]R03942039, [3]R03942071

Abstract

The goal of this project is given social graph, idea adoption records(logs) and idea initiators(test set initial adopters), find subsequent adopters. We explore many models, such as Hottest Recommendation, Matrix Factorization(MF), Singular Value Decomposition, Neural Network, Neural Network with Auto Encoder, and Local Community Detection. Evaluate our methods based on retrieval F-Score.

1. Introduction

The essence of the social influence is a subtle problem combining the human behavior in the social network. When people communicate with each other, they share not only feelings but also value. Today, social network has become indispensable to this generation. Collecting the interaction logs, one can analyze user behavior among social net, and make a profit such as potential advertisement.

In this project, we explore many methods trying to approximate real-life idea influence. We go through TA baseline, Hottest Recommendation comes from statistic analysis, Matrix Factorization and Singular Value Decomposition which is commonly used in recommendation system, Neural Network inspired from Deep Neural Network, and Local Community Detection combines community detection and influence graph.

The organization of this report is as following: Section 2 briefly describes the problem itself and analysis, Section 3 to Section 9 introduces the methods we used, Section 10 reports experiment results.

2. Problem Description

2.1. Definition

Given an directed social graph $G = \{V, E\}$, idea adoption logs L(training data), which comprise of tuples in the form of $(user, idea, timestamp, degree)$, and a set of initial adopters Q(testing data), we want to predict the subsequent adopters A.

$V = \{v\}$ is the set of users in social network, $E = \{(u, v)|u \in V, v \in V\}$ is a set of friendship links in the social network. A record means the user adopts the idea in the timestamp with the degree. The degree range from 0 to 1, representing the acceptance of the user toward the idea. Degree 0.5 is neutral.

2.2. Dataset

We only use the data provided by the course. The graph contains 29053 users with 207759 links. The logs contain 1118671 records with 1000 ideas. There are 3 test cases, each with 10 queries.

Testcase	Total Initiators	Total Adopters
1	3156	7376
2	6073	9118
3	5845	5856

Table 1: 3 test sets

3. Baseline

TA provides a method only based on the graph structure. Simply count the number of friends in the initiators as score, then return the top 100 nodes in the set of users which are no more than 2 step from the initiators. That is, we calculate the score as such:

$$q \in Q, v \in Step2Neighbors(Q) - Q$$
$$score(v, Q) = |\{(v, q) \in E\}|$$

It is obviously weak because the procedure don't use any training data to predict answer.

4. Hottest Recommendation

A song in top list is preferred by almost all people. Inspire by this, a person who is most likely to get influenced is more likely to get influenced again in test set.

We return top 100 users who will easily adopt ideas in the training records exclusive of the initiators. That is, we rank the user score as such:

$$score(v, Q) = |\{(v, i, t, d) \in L\}|, v \notin Q \qquad (1)$$

Scan through the records, we find a set of hottest users, who are easier to adopt an idea. At first sight, it should be the baseline because the procedure is naive and intuitive. But the result is powerful, highly comparable, and will be discussed further.

5. Matrix Factorization (MF)

5.1. Introduction

Advanced music recommendation system uses Matrix Factorization (MF) to find the latent topics inside the user-music preference. The simple way is to construct an $U \times I$ matrix M, where the m_{ij} is the preference of user i toward item j. The preference can be either implicit(e.g. clicks) or explicit(e.g. rating). Figure 1 shows the mathematical structure.

The procedure cluster user based on item preference similarity by the latent topics. Thus, for a new user with limited music preference records, one can model the potential preference probability by matrix multiplication.

5.2. Training and prediction

In this model, time information is not used. We construct the matrix by assigning m_{ij} = degree of user i toward idea j.

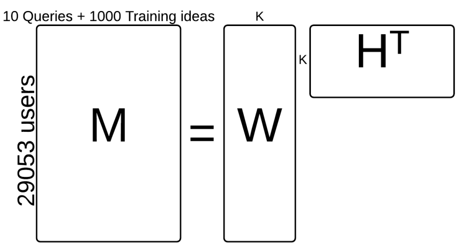

Figure 1: Matrix Factorization.

For m_{ij} not in the training records, we use default value 0. For m_{ij} in test data, we set the initiators 1 and others blank(ϕ), which will be ignored in the training process.

The training of MF is to minimize the L2-regularized L2-reconstruction error:

$$M \approx WH^T$$

$$\min_{W,H} \sum_{m_{ij} \notin \phi} (m_{ij} - \sum_{k=1}^{K} w_{ik} h_{jk})^2 + p \sum_{i,k} w_{ik}^2 + q \sum_{j,k} h_{ik}^2 \tag{2}$$

When prediction, the score of user i toward idea j is calculated by matrix reconstruction:

$$M' = WH^T$$

$$m'_{ij} = \sum_{k=1}^{K} w_{ik} h_{jk}$$

$$score(i, Q_j) = m'_{ij}, i \notin Q_j$$

Then we return the top 100 score exclusive of initiators.

However, We find that the degree of the adoption may not influence the prevalence of the idea, so we simply view all appearing degrees as 1. This heuristic would lead to better performance, which we will discuss further in the Section 10.

6. Singular Value Decomposition

6.1. Introduction

Similar to Matrix Factorization, Singular Value Decomposition is a factorization of a real or complex matrix as well. For an $m \times n$ matrix M, the factorization form should be $U\Sigma V^T$, where the size of U is $m \times m$, the size of V is $n \times n$, and Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal. The diagonal entries of Σ are named as "singular values," and the remain entries are zero. The mathematical structure is shown in Figure 2. Suggest that the Σ matrix has r singular values, we can reduce the memory during calculation by transforming the SVD structure from the upper of Figure 2 to the bottom.

Unlike Matrix Factorization, we don't need to acquire the value of matrix U, Σ and V through training procedure. The columns of U are eigenvectors of MM^T, V's equal to $M^T M$, and singular values are the square roots of eigenvalues of both $M^T M$ and MM^T.

6.2. Prediction

As the same set in the previous section, the entries of M, m_{ij} are the preference of user i toward j. We filled the training

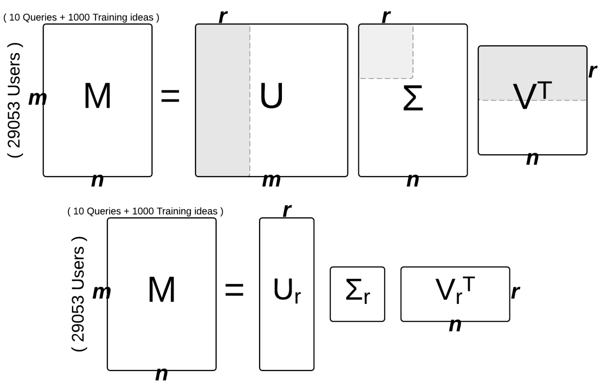

Figure 2: Singular Value Decomposition.

records and testing queries into the corresponding entry m_{ij}. For some entries m_{ij} which are not shown in training and testing data, we consider them neutral and apply 0.5 degree to each one. To the initial adopter in testing queries, we set them 0.8 as well.

After applying Singular Value Decomposition algorithm, we then reconstruct the matrix M and return the top 100 adopters to make prediction:

$$M' = U\Sigma V^T$$

7. Neural Network

7.1. Observation

Information passing among social nets is just like neurons passing messages in Neural Nets. One would get the information most likely from one's friends. In Neural Net, the output of a neuron purely depends on it's former layer neurons.(See Figure 3) Inspired by this idea, we can take each neurons as a person in social net, and train the net by the following method.(Figure 4)

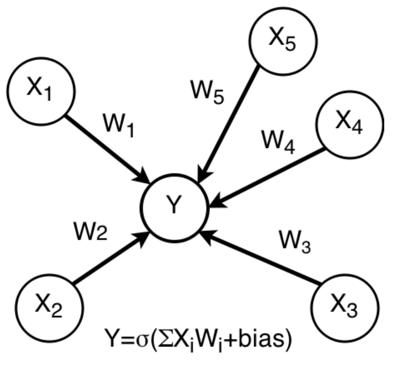

Figure 3: Node Y is affected by Y's 5 friends, X_1 to X_5 with each friend has different affecting weight W_i. There's also a bias in Node Y.

7.2. Architecture

For this Neural Net, the number of neurons in input layer and output layer is equivalent to the number of people in social nets(N in Figure 4), which means the input output neuron representing one person. There's no hidden layer between input layer

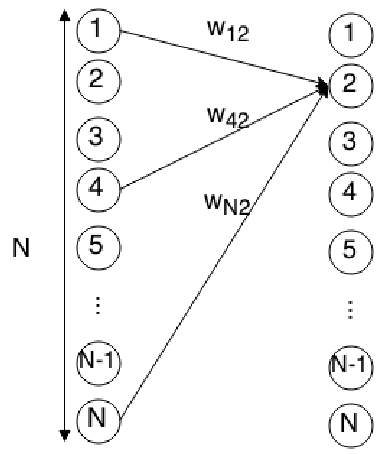

Figure 4: A redraw of Figure 3 to a neural network. In this figure, node 2's friends are node 1, 4 and N. Hence node 2's output only depends on sum of 3 person's weighted degree and bias.

and output layer. Input and output layer are not fully connected. The elements of weight matrix connecting input neuron and output neuron are nonzero if they are friends In Equation 3, y_i is output neuron representing node i in social nets, x_j representing node j in social nets is input neuron with value equaling to it's degree, w_{ji} is non-zero if node j is a friend of node i. For each node i , it has it's own bias term $bias_i$. After summation over all weighted degrees of one's friends and bias, take sigmoid(σ, Equation 4) forcing the output range from 0 to 1 which is the degree of this node.
For input layer, we use binary information to model degree. Default degree is 0, if one's degree is specified, set it's degree to 1 rather than the original value. Because the degree information is noisy, if we simply use binary format, the performance is better.

$$y_i = \sigma(\sum_{j,(i,j)\in E} w_{ji}x_j + bias_i) \tag{3}$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4}$$

7.3. Training and Prediction

During training, the weight is updated only when the connecting user u gets influence from previous adopters. That is, we use degree of friends of u to predict u's degree. Use back-propagation to train the weight matrix.
Prediction is simple, just set the initial adopters to 1, others zero as input vector to Neural Net. Choose top 100 output neurons as answer.

8. Neural Network Auto Encoder

8.1. Motivation

Similar to Matrix Factorization and Singular Value Decomposition, the main idea is to reduce the original large dimension into a small dimension, use this small dimension data to reconstruct the original large dimension. It's called auto-encoding in Neural Networks.

8.2. Architecture

Again, the input layer is degree of each node in social nets, the output ,however, should be exactly the same (see Figure 5). The number of neurons(K) in hidden layer controls the model complexity. Equation 5 is the formula for this Neural Net.

$$\begin{aligned} h_i &= \sigma(\sum_j W1_{ji}x_j + bias1_i) \\ y_i &= \sigma(\sum_j W2_{ji}h_j + bias2_i) \end{aligned} \tag{5}$$

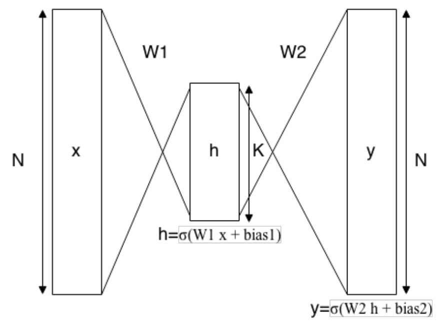

Figure 5: Auto-encoder. x is input vector, y is output vector, h is hidden layer vector. $W1$'s dimension is $K \times N$, $W2$'s dimension is $N \times K$.

x_j is the degree of user j, h_i is a neuron in hidden layer, y_i is the degree of user i. W1 and W2 are matrix, bias1 and bias2 are vectors, these four components are model parameters, which are required to be trained.

8.3. Training and Prediction

Training data is all the ideas regardless temporal information. Each idea is an input to the auto-encoder filled with degrees of all user in the social net, the target output is also the same vector. Default degree is 0, and we use binary information of degree. (If degree is specified, set the degree to 1; otherwise, 0.)
When predicting, initial adopters are turned on (set to 1) others 0. Use this vector as input vector to auto-encoder, sorts the output vector and choose top 100 adopters as answer.

9. Local Community Detection(LCD)

9.1. Introduction

Community structure in social graph can be used to capture some information among users, such as attributes[1]. Traditional community detection technique tries to maximize the ratio between intra-community edges and inter-community edges by expanding the initiators into a bigger set. The procedure would test by adding user one-by-one, only considering the graph structure.

We think there are some community based structure in the graph, but within an influence graph instead of social one only. So we utilize the records to form a influence graph $G' = (V, E', W')$, a directed weighted graph to capture the adoption influence. $E' = \{(u,v)|u \in V, v \in V\}$ and W' is a function that maps the edge (u,v) onto a non-negative real numbers, indicating the influence from u to v.

For all adopters in the training records in same idea, we sort them according to their timestamp (from old to new). Define context window that a node can receive influence, we add this degree to all edge from predecessors in the window to the node. The adoption degree in the records can be discarded(e.g. set to 1) or not. Figure 6 better illustrate the idea on how to do it. The window will slide along time until all the users toward the same idea are seen.

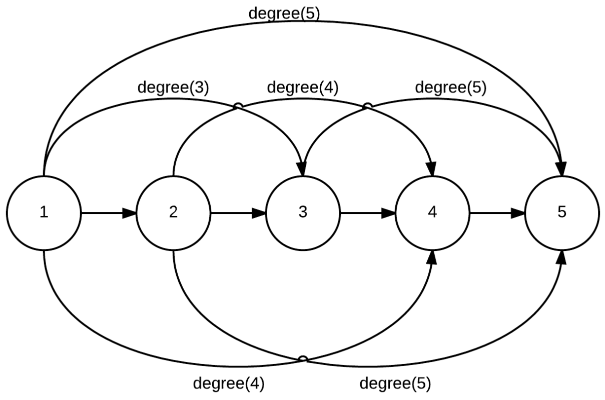

Figure 6: Illustration on how to add the weights onto edges with context window length = 4. Node 1 to Node 5 all adopt some idea sequentially, then we increase edge weights (1,5)(2,5)(3,5)(4,5) by degree of node 5. Any node before node 1 is assumed no influence on node 5.

We repeat the procedure through all the training ideas (1000 ideas). Thus an influence graph can be created.

9.2. Prediction

When prediction, we start the set of initiators as current adopters and find the neighbors, e.g. there is some node in the current adopters has non-negative influence to the node. For each node in the neighbors, we find the node which has most flow-in influence from the current adopters. Then we add the node into the current adopters 1 by 1:

1. current adopters = idea initiators.
2. Find the neighbors, e.g. there is some node in the current adopters has non-negative influence to the node.
3. Find the node among the neighbors which has most flow-in influence summation from the current adopters.
4. Add the node into the current adopters and repeat the step 2 until 100 adopters added.
5. Return the newly added adopters.

We also provide another prediction method, that will predict all the subsequent adopters in one pass:

1. current adopters = idea initiators.
2. Find the neighbors, e.g. there is some node in the current adopters has non-negative influence to the node.
3. Return the top 100 nodes among the neighbors which have most flow-in influence summation from the current adopters.

10. Experiments

10.1. Evaluation

For each method, we report the average recall, precision and f-score on 3 testsets, each with 10 queries. We predict 100 adopters per query.

10.2. Results

We first compare the Baseline, Hottest Recommendation, Neural Network and Singular Value Decomposition in Table 2, and Table 3 shows the Auto-Encoder method using different number of neurons in hidden layer. In Figure 7, we implemented Matrix Factorization in different parameters. The parameters we compared are reduction dimension(k), training iteration(t), value of p and q in equation (2).

We then performed Local Community Detection in the subsequent tables, Table 4, Table 5, and Table 6. We predicted LCD algorithm with and without degree information in the first two tables. Each table also show the scores using different contexts. We found that the more contexts we used, the higher score we acquired, and using binary degree information also beat original degree. Therefore we implemented the LCD in full context, and without degree information in Table 6.

After fine-tuning in every algorithms we used, we shows the best result of each algorithm in Table 7.

Model		Test 1	Test 2	Test 3
Baseline	Precision	0.074	0.054	0.067
	Recall	0.014	0.015	0.014
	F-Score	0.019	0.018	0.022
Hottest Recommendation	Precision	0.388	0.424	0.466
	Recall	0.140	0.114	0.119
	F-Score	0.174	0.145	0.176
Neural Network	Precision	0.163	0.162	0.132
	Recall	0.048	0.043	0.031
	F-Score	0.062	0.054	0.048
Singular Value Decomposition	Precision	0.095	0.133	0.139
	Recall	0.0413	0.015	0.024
	F-Score	0.023	0.026	0.041

Table 2: Baseline, Hottest Recommendation, Neural Network and Singular Value Decomposition.

10.3. Discussion

Neural Network is not as good as we expected. The F-Score of Neural Network is worse than Hottest Recommendation. Actually, any model should do better than Hottest Recommendation, since Hottest Recommendation simply gives statistic result. The reason may be over-trust graph information, and small training data.

Singular Value Decomposition also shows poor performance, and it is only slightly better than the Baseline. We conclude that the assumption of neutral adopters leads to the poor evaluation score. In the SVD architecture, we assumed the entries that is not mentioned in testing queries to be 0.5, which might affects the predicting result. Unlike SVD algorithm, we can assume these unseen adopters to be missing values in Matrix Factorization so that we could learn these values during training procedure without adding noises.

In Neural Network Auto-Encoder (Table 3), the result is quite interesting. First, the F-score is higher than Hottest Rec-

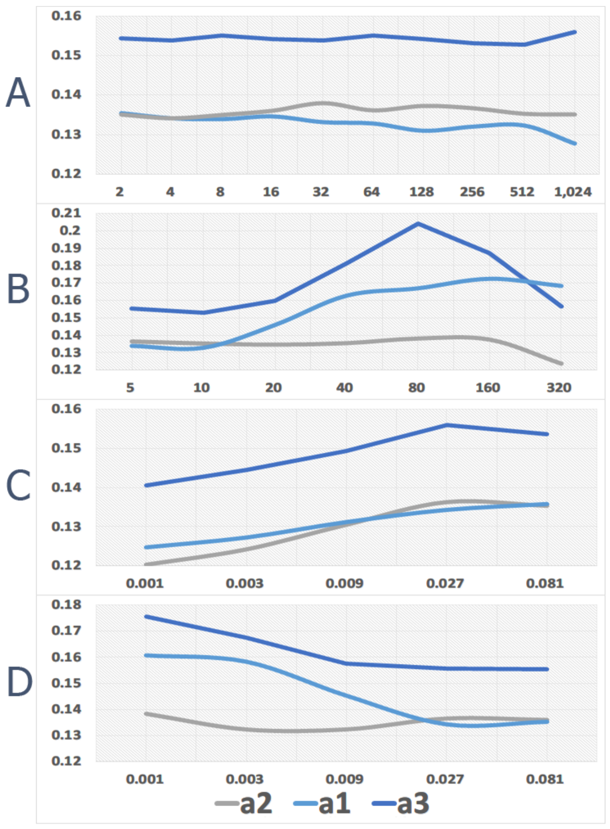

Figure 7: Matrix Factorization using different parameters in F-measure score. (A) k from 2 to 1024 with t=8, p=0.05, q=0.05 (B) t from 5 to 320 with k=32, p=0.009, q=0.001 (C) p from 0.001 to 0.081 with k=32, t=40, q=0.027 (D) q from 0.001 to 0.081 with k=32, t=40, p=0.027

Model		Test 1	Test 2	Test 3
K=50	Precision	0.386	0.425	0.466
	Recall	0.140	0.114	0.119
	F-Score	0.174	0.145	0.176
K=100	Precision	0.413	0.501	0.503
	Recall	0.144	0.118	0.124
	F-Score	0.179	0.152	0.185
K=200	Precision	0.419	0.474	0.543
	Recall	0.146	0.116	0.128
	F-Score	**0.182**	**0.147**	**0.192**
K=300	Precision	0.414	0.483	0.542
	Recall	0.146	0.116	0.127
	F-Score	0.181	0.148	0.191
K=400	Precision	0.410	0.481	0.528
	Recall	0.144	0.118	0.126
	F-Score	0.179	0.150	0.189

Table 3: Auto-Encoder with number of neurons(K) in hidden layer changing.

ommendation, which means the model learns more than statistic analysis. Second, as K(number of neurons in hidden layer) decrease to about 50, the scores are close to Hottest Recommendation. If K equals to 0, the result is exactly Hottest Recommendation, since the output layer is only depends on the bias term in the output layer. Third, as K increases, the computation

Model		Test 1	Test 2	Test 3
Context = 1	Precision	0.349	0.355	0.297
	Recall	0.121	0.095	0.079
	F-Score	0.152	0.119	0.117
Context = 5	Precision	0.362	0.356	0.324
	Recall	0.126	0.096	0.087
	F-Score	0.158	0.122	0.129
Context = 10	Precision	0.360	0.352	0.322
	Recall	0.125	0.095	0.087
	F-Score	0.156	0.121	0.129
Context = 50	Precision	0.370	0.430	0.452
	Recall	0.139	0.109	0.114
	F-Score	0.165	0.141	0.169
Context = 100	Precision	0.374	0.449	0.471
	Recall	0.125	0.114	0.117
	F-Score	**0.159**	**0.147**	**0.175**

Table 4: Local Community Detection based on influence graph with degree and 1-by-1 prediction.

Model		Test 1	Test 2	Test 3
Context = 1	Precision	0.375	0.422	0.448
	Recall	0.131	0.112	0.113
	F-Score	0.165	0.143	0.169
Context = 5	Precision	0.386	0.429	0.467
	Recall	0.138	0.112	0.119
	F-Score	0.172	0.144	0.177
Context = 10	Precision	0.389	0.428	0.470
	Recall	0.140	0.113	0.119
	F-Score	0.175	0.145	0.177
Context = 50	Precision	0.396	0.443	0.482
	Recall	0.141	0.117	0.121
	F-Score	0.176	0.149	0.181
Context = 100	Precision	0.389	0.460	0.496
	Recall	0.133	0.120	0.126
	F-Score	**0.168**	**0.153**	**0.187**

Table 5: Local Community Detection based on influence graph without degree(e.g. whenever a adopter occurs, add the influence edge by 1) and 1-by-1 prediction.

Model		Test 1	Test 2	Test 3
1-by-1 Prediction	Precision	0.387	0.469	0.569
	Recall	0.105	0.090	0.127
	F-Score	0.139	0.124	0.192
One Pass Prediction	Precision	0.442	0.530	0.580
	Recall	0.140	0.177	0.136
	F-Score	**0.177**	**0.163**	**0.204**
One Pass Prediction Time Decay = 0.99	Precision	0.299	0.277	0.363
	Recall	0.103	0.082	0.104
	F-Score	0.128	0.103	0.151

Table 6: Some comparison about Local Community Detection based on influence graph in full context without degree.

time does increase linear to K, and the scores do not monotonically increase. The Highest score is around K = 200 due to the training data size and the problem complexity.

While examining Matrix Factorization techniques, we compared the F-measure score using different parameters. The results shown in Figure 7 can be discussed separately. In Figure 7-(A), we discover that the value of dimension shares low corre-

Model		Test 1	Test 2	Test 3
Baseline	Precision	0.074	0.054	0.067
	Recall	0.014	0.015	0.014
	F-Score	0.019	0.018	0.022
Hottest Recommendation	Precision	0.388	0.424	0.466
	Recall	0.140	0.114	0.119
	F-Score	0.174	0.145	0.176
Matrix Factorization	Precision	0.476	0.438	0.568
	Recall	0.139	0.108	0.137
	F-Score	**0.178**	**0.139**	**0.206**
Singular Value Decomposition	Precision	0.095	0.133	0.139
	Recall	0.0413	0.015	0.024
	F-Score	0.023	0.026	0.041
Neural Network	Precision	0.163	0.162	0.132
	Recall	0.048	0.043	0.031
	F-Score	0.062	0.054	0.048
Auto-Encoder	Precision	0.419	0.474	0.543
	Recall	0.146	0.116	0.128
	F-Score	0.182	0.147	0.192
Local Community Detection	Precision	0.442	0.530	0.580
	Recall	0.140	0.177	0.136
	F-Score	**0.177**	**0.163**	**0.204**

Table 7: Evaluation after fine-tuning in every algorithm.

lation with the performance, and Figure 7-(B) shows the highest score if we train the model for 80 iterations. We then find that the higher value of p we chose, the higher score will we get. However, parameter q shows the opposite potential.

In Local Community Detection, the last method we proposed, we first set an experiment about using the degree information or not. The results shown in Table 4 and Table 5 are obviously that the LCD algorithm will perform better if dropping out degree information. Besides, if we consider more context, the accuracy will also increase. Therefore, we perform LCD considering the full context in Table 6 using different variables. In the first two rows, we find out that ”One Pass Prediction” method beats the ”1-by1 Prediction”. We also tried to set the influence with temporal decay, which means the longer distance between two adopters, the lower influence they get.

After fine-tuning parameters in every algorithm, we select the highest scores and make comparison. The result are shown in Table 7. Local Community Detection algorithm outperform other methods.

11. Conclusion

Different models perform well in different cases. It is proved that the LCD can better catch the temporal information with full context, and influence graph can predict the subsequent adopters very well. However, we do neglect the degree in the training logs to achieve better results, because the prevalence of an idea adoption should not depend on the degree of the adoption. We initialize the influence graph with actual social graph structure, but the significance is a bit slight. After the exploration on various models, we propose the LCD as our final recommendation.

12. References

[1] *You are who you know: Inferring user profiles in Online Social Networks.* Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel In Proceedings of the 3rd ACM International Conference of Web Search and Data Mining (WSDM’10), New York, NY, February 2010.

[2] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin. ”A Fast Parallel Stochastic Gradient Method for Matrix Factorization in Shared Memory Systems.” Technical report 2014.

[3] *ImageNet* `http://www.image-net.org/`