Report in Data Mining - Analyzing Hypergraphs

Oskar Gross, Anna Leontjeva

January 28, 2010
Abstract

Nowadays social networking and large user-based applications have become more and more popular. Also, this kind of applications can easily be used by users on wrong purposes as it is impossible to analyze the actions of these users manually. This means that there is a reasonable need for automated application which can find users who may be harmful for other users or for the business. In our project the goal was to find interesting patterns of individual behaviour in a large network. In this report we describe how we analyzed a subset of a large network, what kind of applications we implemented and we also demonstrate our results.
Chapter 1

Introduction

1.1 Aim

The customer graph encodes various relations between different customers in the business network. The main aim of the project is to find interesting components or structures in a large graph. We did a descriptive analysis of the customer data and tried to find patterns which could somehow show us outliers of usual behaviour. The outcome of this knowledge is that we can classify a typical account into a “interesting behaviour of a group” or into “normal user” - basically a possibility to detect unusual customer behaviour and understand what are the patterns of interesting acts and how we can detect them from a large network structure.

1.2 Definitions

For the start we give some definitions in order to give some intuition about the data which we describe in the next section.

**Definition 1.** Hyperedge is a connection between two or more vertices of a hypergraph. A hyperedge connecting just two vertices is simply a usual edge.

**Definition 2.** The colour of the hyperedge represents an action made by a user. All users who have made the same actions with the same parameters belong to the same hyperedge.

**Definition 3.** Hypergraph $H$ is a pair $(V(H), E(H))$ where $V(H)$ is the set of vertices of $H$ and $E(H)$ is a family of subsets of $V(H)$ called the hyperedges of $H$

Less formally - all vertices in a hyperedge are connected to each other with each other and hypergraph is the set of hyperedges.[1] This structure gives an opportunity to represent a very large graph in quite a compact manner which we can also see from Figure 1.1.
1.3 Data

During the project we had two kinds of data - the generated data and generated data which should somehow describe the “not-so usual” users. To be mentioned, further we will use the term “dummydata” for data. Dummydata was generated in order to deliver us something similar to the real data, so that we can make analysis on similar data before the real one is handed over. As the “dummydata v1” was not very interesting, in this report the results are accomplished on the second generated data. Unfortunately the interpretations in this report are quite “wild guess” as we do not have data of the most usual behaviour.

1.3.1 Structure

The network structure was given as a hypergraph - which means that for each vertex we know the edges in which the vertex belongs to. The textual representation can be found from Table 1.1. There are four different attributes of one transaction - vertex_id, edge_weight, edge_type and edge_id. Each of the attributes have following properties:

- *vertex_id* attribute specifies a customer.
- *edge_weight* attribute describes a connection strength of the vertices in the edge and it takes three values 1, 2 and 3. Value 1 indicates that the connection is very weak or even does not exist, value 2 shows that there is a weak connection between two accounts (for instance account usage from
the same internet cafe), and value 3 indicates that the two persons have had real contact between each other.

• *edge_type* attribute takes values from 1 - 8. The value shows how these two accounts are connected - whether it was the same ip-address or these users used the same email address.

• *edge_id* refers to a specific edge. All vertices which are connected to the same edge have connections between them.

<table>
<thead>
<tr>
<th>vertex_id</th>
<th>edge_weight</th>
<th>edge_type</th>
<th>edge_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x_1$</td>
<td>$y_1$</td>
<td>$z_1$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>$x_n$</td>
<td>$y_n$</td>
<td>$z_n$</td>
</tr>
</tbody>
</table>

Table 1.1: Hypergraph representation
Chapter 2

Data Analysis

In this chapter we show, what kind of analysis we did on the data and what were the results.

2.1 Sampling

In this report the results given are made on relatively small dataset - 2194 transactions were generated from the data generator. The reason for the size is quite simple - the generated data structure is completely different from the previous data. The calculations of 2194 transactions of “dummydata v2” graph took the same time as 30000 transactions of the generated data. This problem could have been solved, but unfortunately we got the “dummydata v2” data generator just a few days before this report, and we did not manage to adapt with the situation. We will give explanations from the software point of view in the end of this report.

The strange number (2194) of transactions sampled is caused of the fact that during sampling we needed to extract all information about certain user. If we had generated 2000 transactions, for instance, then we would have had 1 user with missing data and this could have brought us to biased results.

Regarding these facts these results and findings still have some value as we still get some overview of this data and we can make some hypothesis of the big dataset.

2.2 Visualization

It seems reasonable to first take a look at the data and then compute. As the hypergraph visualization would have needed custom implementation we decided to create an edgegraph and visualized it with igraph software packet.[2][4]

Definition 4. Edge graph is hypergraph representation as a graph. Each hyperedge is translated into vertex and an edge between these two translated edges
is added if they shared 1 or more vertices in the hyperedge. The edge weight between these two edges is the number of common vertices.

The visualization of the edge graph can be seen on Figure 2.1. Current figure does not give very much information to us - it seems that the graph has many edges and is strongly connected, but that is all information we can extract.

![Figure 2.1: Edgegraph visualization](image)

2.2.1 Connected Component Visualization

Another interesting way to visualize a graph is to make different connected components of different colour, the result of this technique can be seen in Figure 2.2. From the figure we see that we have big and small connected components. If we need to interpret this somehow, then it might mean that the components are interesting groups.
2.3 Graph Structure

From the Figure 2.2 - we saw that the graph consists of both - big and small connected components. Now that we see that the graph has many communities we would also like to take a look at other descriptors.

2.3.1 Vertex to Vertex Path Lengths

One way of getting some idea of the graph structure is finding the graph diameter - the longest shortest path from one vertex to another. As we are dealing with a large graph it does not seem reasonable to find the diameter of the graph as it takes very long time to compute. Additionally we do not lose much information
if we just took a reasonably big number of random vertices and calculated
distances between them. On the Figure 2.3 we can see the histogram of the
distance distribution of 1000 random path lengths.

First thing which we can see is that most of the random vertices taken
are connected to each other. In 241 cases there was no connection between two
vertices and in other cases the vertices were connected. Also we can see, that the
length between two connected vertices was around 4-6 edges and the frequency
of path length drastically falls if it is more than edges 10 long.

![Histogram of distances](image)

Figure 2.3: “Dummydata v2” graph edge graph histogram of distances

Now we can also make short overview of the path lengths which were greater
than \(-1\) (see Figure 2.4). Basically we see the same thing as on the histogram
before - most of the values are ranging around 5 and there is sharp fall for
distances more than 10.
2.3.2 Vertex degrees

Definition 5. *The vertex degree of a graph shows how many edges are incident to the vertex, with loops counted twice. In our work we use the vertex degree as a measure to show how many neighbours does the given vertex have.*

Definition 6. *In the hypergraph sense for given vertex the vertex degree shows into how many hyperedges the vertex belongs to.*

As the graph is very large, it is quite impossible to give reasonably accurate estimation of the graph connectivity. We also saw this on Figure 2.2. Taking this into account we decided to find the vertex degrees for both - hypergraph and graph converted from hypergraph. Additionally for every degree we also found what are the vertex degrees for each type which shows how many edges with current type are incident to current vertex.

First of all lets take a look at the vertex degrees of hypergraph sense. The vertex degree distribution for hypergraph can be found in Figure 2.5 on the left. As we can see, most of the vertices have degree less than 20 but as we can see, there are some outliers. To get a better overview of these small values, in the Figure 2.5 on the right we see the same distribution without outliers. As we can see most of the vertices belong 3-5 edges. This also explains the graph which we saw on the picture 2.2 - we have a lot of vertices which belong to many edges, a couple of vertices which belong to 1 or 2 edges.

Now we should take a look at the vertex degrees of the converted graph. In the Figure 2.6 we can see the distribution of vertex degrees. We see that the half of verte neighbours is ranging around 10 – 69 and most of the vertex degrees are greater than 65.
Figure 2.5: Hypergraph vertex degree distribution with and without outliers

Figure 2.6: Graph vertex degree distribution
2.3.3 Vertex Degrees by Type

**Definition 7.** The typed vertex degree of a graph shows how many edges of specified type are incident to the vertex, with loops counted twice. In our work we use the typed vertex degree as a measure to show how many neighbours with current type does the given vertex have.

**Definition 8.** In the hypergraph sense for given vertex the typed vertex degree shows into how many edges of specified type the vertex belongs to.

As different edge types mean different actions from user we also decided to take a look at the vertex degrees by type. By this information we can see if some types of actions are more represented in “dummydata v2” group than in usual groups. In the Figure 2.7 we can see that types 2, 4 and 5 are more represented than others. If we knew actions behind these different types, we maybe could somehow interpret this, but at the moment we can say, that it just is so.

In the Figure 2.8 we can see the hypergraph vertex degree distribution and the edge size distribution by type. We can see that most of these vertices belong to only one edge with specific type. What maybe very interesting is the fact that for type 6 there are 3 very interesting outliers - once again stating that if we knew what the different types mean, we could interpret the data much better. One interpretation maybe that the outlier which is connected to 300 edges with type 6 maybe some kind of central user who is in the center of all these interesting transactions.

From the edge size distribution we can see, that the hyperedges with type 6
are quite small and hyperedges with types 2, 4 and 5 have some outliers and are a little bigger than others. As we are dealing with data which may describe the interesting behaviour of users this could mean, that these types of actions are more relevant than other types.

2.3.4 Vertex Type Correlations

For this we found correlation table which can be found from Figure 2.9. From the correlation table we can see how correlated the types are. From the correlation table we can see, that the vertex degree is strongly correlated with vertex degree of type 2, 4 and 5. Interesting fact is that vertex degree type 2 is strongly correlated with type 5. Basically the correlations mean that if degree of one edge type is increasing then the degree of other edge type is also increasing.

Once again knowing the background data behind these types would make it easier for us to interpret these correlations more, but this again might show us what actions are involved.
2.4 Programs

During the project we wrote three different programs. In the beginning we understood that we will have to analyze a very large graph and because of this we implemented a SGraph program in C++. The SGraph’s is a data converter - it reads in the file which was generated and it is possible to convert the hypergraph into various formats for later statistical analysis. Also the processes which need a lot of computations are implemented in SGraph - for instances path lengths between vertices, building the edge graph et cetera.

So as we had a workshop in the company we decided to write the C++ program into R[3] to ensure that the applications will work on other computers and this is the second program we implemented.

2.4.1 Scalability

It is important that the programs should be able to work also on large graphs. At the moment the R script does not handle very large graphs and the C++ program needs a little bit of tweaking, to get it to work with the descriptive analysis program written in R. The program written in C++ can handle relatively large graphs, another positive side of the SGraph program is memory usage - for a 198 MB (which is around 10 000 000 transactions) file it reserves memory for 398 MB.

Figure 2.9: Vertex degree correlations
Chapter 3

Conclusion

3.1 Discussion
In our work we presented the methods we used to analyze hypergraph data. We showed different manipulations which can be done with hypergraphs and also we have some ideas how to interpret this kind of data. We saw that at the moment we have quite reasonable programs written and most of the meaningful results will be available after the additional data will be received. Also we showed that we have scalable applications for further analysis with much more bigger datasets.

3.2 Future Work
At the moment we have quite good ideas and tools for further research of this kind of data. Unfortunately all the meaningful results we can give after we have used it on real data. Maybe that the results we have found today already give us something but, maybe it is just a little step forward. In the future there we also plan to synchronize the data types used in R and SGraph, also frequent components mining could be quite useful. We would also like to analyze the graph so that we would divide data into three groups - edges with weights 1, 2 and 3, where the weights show the connection strength between the user and the edge.
Bibliography


