Maximal Bi-clique Extraction Problem

July 2014

Indice

1 Introduction

In this document we will discuss about an algorithm due to finding approximate Maximal Bi-clique in a bipartite graph.

We use a clustering method as a first step, and assuming it is good enough, we try to refine it by locating and eliminating impurities and errors. Since the first cluster finding algorithm will not be inspected, the hypotheses and the assumption made in this paper may differ from the real data.

Our main goal is to locate a Maximal Bi-clique whit an high number of vertices on one side of the graph. More precisely, we set a least bound (called *minsupport*), on the left side, and cluster the nodes so that there's a high probability of extracting a bi-clique that respects the bound.

The two sides of the graph will be called **Itemset** and **Item**, since the original formulation of the problem involved transactions and items, so an edge between two elements can be read as "the item X was purchased in the transaction Y", or, symmetrically, "The transaction Y contains the item X".

2 Algorithm

2.1 Step 1: Clustering

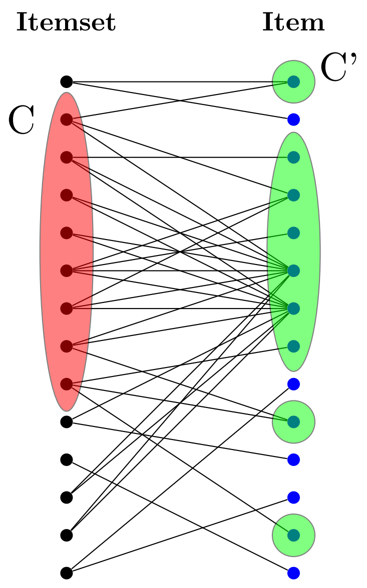
On the first step, we use a clustering algorithm, based on hash functions, in order to detect a group of itemsets which have a lot of common items. In the graph, it can be read in terms of neighbours, in fact the selected nodes will have many neighbours in common.

We make the assumption that this first selection of nodes (called C) is a good approximation of the left side of a maximal bi-clique, or, to be more precise, it has few:

impurity itemset in C which has a small intersection with the elements of C

omission itemset not belonging to C which has a big intersection with the elements of C

if n is the cardinality of C, we also request that $n >> minsupport$, so that deleting the impurities in C do not reduce the number of nodes chosen below the lower bound *minsupport*



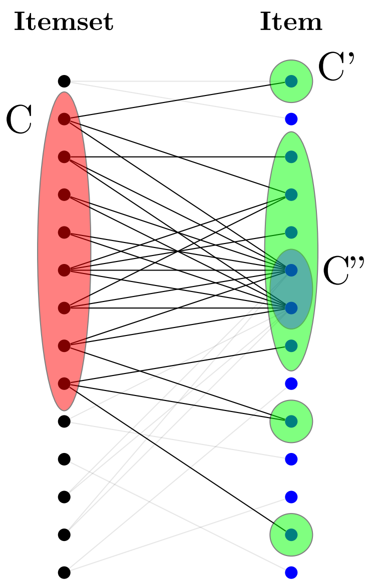
2.2 Step 2: Neighbours

On the second step, we consider the neighbours of C, defined as the nodes which are connected to at least one node in C, and we call this set C'.This is also the union of the itemset in C, thus is usually bigger then C.

Our goal is to find a subset of C' such that every node in it is connected to a lot of nodes in C, so its cardinality is way smaller than the ones of C and C'. This time, there's no bound on the chosen subset, but the items are most likely already clustered, so we can use the degrees of the nodes in C' to extract information about the graph.

2.3 Step 3: Threshold on Degree

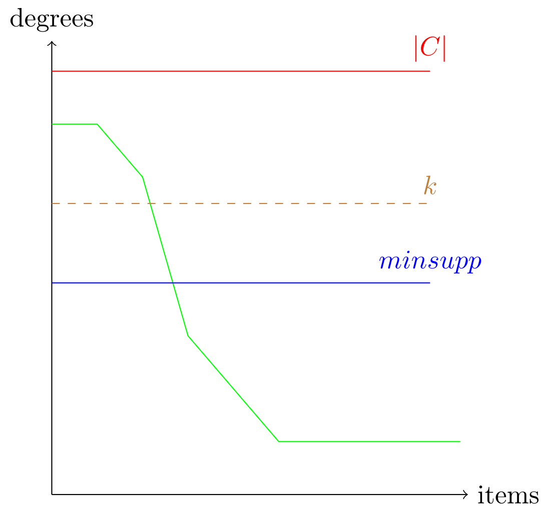
For the third step, we consider the bipartite subgraph that has $C \cup C'$ as nodes, with the associated edges. Then, we study the degree distribution of the nodes in C', and isolate the nodes with the highest degrees.



To do so, we have to set a threshold on the degrees of the nodes, that has to depend, in general, by a lot of factors. Under the assumption that the first clustering operation was good enough, however, we can hypothesize that the distribution has some peculiar properties.

For example, the nodes with the highest degrees will have close degrees, not so far from the cardinality of C, whereas the majority of the nodes will have very low degrees, far below the *minsupport* constant, and will create a *long tail* in the graphic of the degrees. This phenomenon is caused by the fact that the itemsets in C have not many item in common, but is also amplified by the presence of the *impurities* in C.

However, the main characteristic of the degree graph is the big difference between the nodes with a high degree, and the ones with a low one, and the nearly inexistence of nodes with intermediate degrees.



Finding a good threshold value k depends by the distribution, but it has to follow the inequality

$$|C| > k > minsupport$$

One of the possible k is the mean between the maximum of the degrees and the value *minsupport*, but its efficiency has yet to be tested on real data, so we won't discuss further about this.

Called C'' the subset of C' chosen by the threshold, let's finally find our biclique.

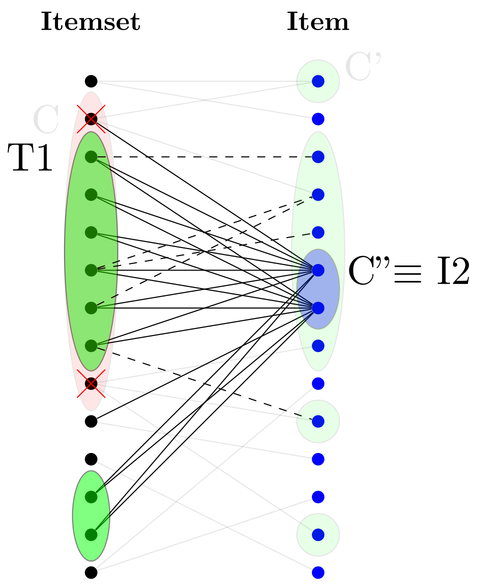
2.4 Step 4: Eliminate Errors

For step four, we focus on C'' and its neighbours.

Taken a node v, and called $N(v)$ the set of its neighbours, we define

$$T1 = \bigcap_{v \in C''} N(v)$$

the itemsets which contains all the item in C'', or, equivalently, the nodes on the left side which are connected to all the nodes in C''. $T1$ will be very similar to the set C, but it will contain the *omissions*, and it will eliminate the *impurities*, since the first have a lot of item in common with C, so probably they contain all the items in C'', and the second don't.



Since we assume that the first step made few errors, the cardinality of $T1$ is close to that of C, but in general is far smaller, so it's important that $|C| >> minsupport$, so that the cardinality of $T1$ doesn't fall below our bound

An other reason for considering $T1$ is because the subgraph composed by $T1$ and C'' is obviously a bi-clique, and in most of the cases, it's already maximal. In fact, we can observe that $T1$ can't be expanded further without adding itemsets which don't contain all of C''. Moreover, if there was a node outside C'' that is connected to all nodes in $T1$, it would already be in C', and it would have an high degree, so it's strange that the threshold had discarded it.

However, we can't be sure it wouldn't happen, so we have to resort to a last stage

2.5 Step 5: Maximal Bi-clique

In this last step, we construct the right side of our maximal bi-clique by the intersection of the itemsets in $T1$. In fact, in the notation of the precedent step, we define

$$I2 = \bigcap_{v \in T1} N(v)$$

so that $(T1, I2)$ form a bi-clique.

It is maximal since we can't expand neither $T1$ or $I2$. In fact $I2 \subseteq C''$, so if we can now expand $T1$ we could have done it in step 4, absurd; and, if we can expand $I2$, it wouldn't be the intersection of the itemsets in $T1$.

3 Conclusion

This algorithm is still not complete, and it needs several tests on one hand for proving the soundness of the assumptions made, and on the other hand for a more precise analysis of the threshold and the distribution in step 3.

Thus the next work will be an analysis of the clustering algorithm and its effects on real data.