

CS599: Structure and Dynamics of Networked Information (Spring 2005)
02/09/2005: Correlation Clustering
Scribes: Ashish Vaswani

So far, we have mostly talked about communities in the sense of discovering one, or a few, densely linked subgraphs. We departed from this interpretation at the end of last lecture, when we defined the notion of the *modularity* of a clustering. There, we are interested in the division of a graph into disjoint partitions (or clusters) of nodes, and the quality of this clustering. Clustering of data, mostly in metric spaces, is one of the most well-studied problems in CS, mostly due to its applications to machine learning and classification.

Here, we look at the relatively new concept of *correlation clustering* [1]. In correlation clustering, each edge of the graph is annotated with a label of '+' or '-', expressing that the two endpoints were observed to be similar or dissimilar, respectively. The goal is then to find a clustering that puts many '+' edges inside clusters, and '-' edges between clusters. However, these goals may be conflicting, as can be seen for a triangle with two edges labeled '+' and one labeled '-'. Notice that the number of clusters is not pre-specified in this problem.

This notion of clustering can be useful when we can identify if a link constitutes an endorsement, or the opposite. For instance, in many competitive scenarios (for instance, politics or sports), pages will link to other pages with the explicit goal of deriding the content. This can be frequently identified from anchor text and similar clues. In this sense, correlation clustering may help us in identifying communities with aligned interests, which compete with other communities.

More formally, given the graph $G = (V, E)$ on n vertices, we write $+(i, j)$ if the edge between i and j is labeled '+' and similarly for $-(i, j)$. The optimization problem can now be expressed in two ways:

- Maximize the number of *agreements*, i.e., the number of '+' edges inside clusters plus the number of '-' edges across clusters.
- Minimize the number of *disagreements*, i.e., the number of '+' edges across clusters plus the number of '-' edges inside clusters.

Clearly, the solution optimizing the first criterion is the same as the one optimizing the second. However, we will see that the two objective functions differ with respect with how well they can be approximated.

The correlation clustering problem is NP-hard, even for the complete graph (where each possible edge exists, and is labeled either '+' or '-'). Hence, we are interested here in approximation algorithms.

1 A Simple Algorithm for Maximizing Agreements

For the maximization version, we may choose to go just after one of the two types of edges. By putting all nodes in one cluster, we get all the '+' edges right — by putting each node in its own cluster, we get all the '-' edges right. This suggests the following algorithm [1]:

If the number of '+' edges in the graph is larger than the number of '-' edges, then put everything into one cluster, else put every node in its own cluster.

Claim 1 *This is a $\frac{1}{2}$ -approximation.*

Proof. If the graph has m edges, then the optimal solution can have at most m agreements. Our algorithm produces a clustering that has at least $\frac{m}{2}$ agreements. Hence, it is a $\frac{1}{2}$ -approximation. ■

One consequence is that in a complete graph with assignments of edges, there must exist a clustering with at least $\frac{n(n-1)}{4}$ agreements (half of the total number of edges in the complete graph).

1.1 Can we do better?

While the algorithm is a $\frac{1}{2}$ -approximation, it is hardly satisfactory in a practical sense. We don't need a new clustering model or algorithm if all it does is lump everything together, or put each node in its own cluster. So naturally, we want to know if the approximation guarantee can be improved.

In [1], the authors develop a PTAS (*Polynomial Time Approximation Scheme*) for the maximization of agreements version in a complete graph. They present a class of algorithms, parameterized with some $\epsilon > 0$, such that the algorithm with parameter ϵ is a $(1 - \epsilon)$ approximation with running time $O(n^2 e^{O(1/\epsilon)})$. While this grows exponentially in ϵ , for any fixed ϵ , the algorithm takes polynomial time.

For the minimization version, we cannot hope for a PTAS, as shown by the following theorem [1].

Theorem 2 *The minimization version is APX-hard even for complete graphs. That is, there is some $\alpha > 1$ such that the minimization version cannot be approximated within α , unless $P = NP$.*

2 A Four Approximation for minimizing disagreements in complete graphs

Given that the maximization version is (at least theoretically) settled for complete graphs, we next look at the minimization version. [2] gives a 4-approximation based on rounding the solution to a Linear Program.

The idea of the Linear Program is to start with an Integer Program. For each pair i, j of nodes, we have a variable x_{ij} which is 1 if they are in different clusters, and 0 otherwise. To make these variables consistent, we have to require that if i and j are in the same cluster, and j and k are in the same cluster, so are i and k . This can be expressed by saying that $x_{ik} \leq x_{ij} + x_{jk}$. Subject to this consistency requirement, we want to minimize the number of '+' edges between clusters, plus the number of '-' edges within clusters. Thus, we obtain the following IP.

$$\begin{aligned} \text{minimize} \quad & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) \\ \text{subject to} \quad & x_{ik} \leq x_{ij} + x_{jk} \quad \text{for all } i, j, k \\ & x_{ij} \in \{0, 1\} \quad \text{for all } i, j \end{aligned}$$

As we know, solving IPs is itself NP-hard, so as usual, we want to relax the IP to a linear program, by replacing the last constraint with the constraint that $0 \leq x_{ij} \leq 1$ for all i, j . This LP can now be solved; however, the output will be fractional values x_{ij} . Our goal is to somehow convert these fractional values into integer ones ("round" the solution) in such a way that the objective function value of the rounded solution is not much more than that of the fractional solution. As the latter is a lower bound on the optimum integer solution, we will derive an approximation guarantee.

After obtaining the fractional x_{ij} values (in polynomial) time, we need some intuition for dealing with them. We notice that the main constraint is just the triangle inequality. Hence, if we define $x_{ii} := 0$ for all i , the x_{ij} exactly form a metric space. In other words, the optimum solution is the metric minimizing the objective function. Nodes that are close in the metric space can be considered as being "almost in the same cluster", while distant nodes are "quite separated". So we likely want our clusters to consist of nodes that are mutually close. At the same time, we need to choose the boundaries carefully, so as not to cut too many edges. It turns out that the following algorithm 1 from [2] carefully trades off between the different types of costs:

Claim 3 *This is a 4-approximation.*

Proof. The idea of the proof is to compare the mistakes that are incurred with the above algorithm against the LP cost, i.e., by showing that whenever a cluster is formed, the number of mistakes ('+' edges across clusters plus '-' edges inside clusters) is at most four times the corresponding LP cost of the corresponding edges.

First, by simple applications of the triangle inequality and reverse triangle inequality, we obtain the following relationships between the distances (which will be used later).

Algorithm 1 LP-Rounding

```
1: Start with a set  $S$  containing all the nodes.
2: while  $S \neq \emptyset$  do
3:   Select an arbitrary node  $u \in S$ .
4:   Let  $T := \{i | x_{ui} \leq \frac{1}{2}\} \setminus \{u\}$ 
5:   if Average distance from  $u$  to the vertices in  $T$  is at least  $\frac{1}{4}$  then
6:     Let  $C := \{u\}$  (a singleton cluster).
7:   else
8:     Let  $C := T \cup \{u\}$  (a cluster).
9:   end if
10:  Remove  $C$  from  $S$ .
11: end while
```

Lemma 4 For any nodes i, j, u :

- The cost of a '+' edge (i, j) incurred by the LP is at least $x_{ij} \geq x_{uj} - x_{ui}$.
- The cost of a '-' edge (i, j) incurred by the LP, is at least $\max(0, 1 - x_{uj} - x_{ui})$.

We will show the claim for each cluster separately. So let u be a cluster center of the cluster C . We can then sort the other nodes by increasing distance from u , i.e., whenever $x_{ui} < x_{uj}$, then we say that $i < j$ (ties are thus broken arbitrarily).

We distinguish between the two cases when a singleton cluster is created, or when the cluster includes T .

Singleton cluster $\{u\}$

This case occurs when the average distance from u to the vertices in T is at least $\frac{1}{4}$. Our algorithm then pays for all the '+' edges incident with u . For each such '+' edge whose other endpoint is not in T , the LP pays at least $\frac{1}{2}$. And for the edges whose other endpoint is in T , the LP pays at least

$$\sum_{i \in T, +(ij)} x_{ui} + \sum_{i \in T, -(ij)} (1 - x_{ui}) \geq \sum_{i \in T} x_{ui} \geq \frac{|T|}{4}.$$

Therefore, for a singleton cluster, our algorithm pays at most four times the LP cost.

Non-singleton clusters

In this case, the algorithm could be making two kinds of mistakes:

1. Allowing '-' edges inside C .
2. Cutting '+' edges between C and $S \setminus C$.

Here, we analyze the costs of negative edge mistakes. Positive edge mistakes are covered in the next lecture.

If we have a '-' edge (i, j) such that both i and j are close to u , then they must be close to each other, so the fractional solution must also pay a lot for this edge. Specifically, if x_{uj} and x_{ui} are both at most $\frac{3}{8}$, then from the second part of Lemma 4, x_{ij} is at least $1 - x_{uj} - x_{ui} \geq \frac{1}{4}$. So the cost for any such edge incurred by our algorithm is at most 4 times that of the LP solution.

For the remaining '-' edges, we will not be able to come up with a bound on an edge-by-edge basis. Indeed, if two nodes i and j are at distance $\frac{1}{2}$ each from u , they may be at distance 1 from each other, so the fractional solution incurs no cost. Instead, we will compare the costs node by node: specifically, we will show that for any node j , the number of '-' edges between it and nodes i that are closer to u than itself is at most four times the corresponding cost of the LP solution.

So we fix a node j with $x_{uj} \in (\frac{3}{8}, \frac{1}{2}]$. The total cost of edges (i, j) with $i < j$ (both '+' and '-' edges) that the LP solution incurs is

$$\sum_{i < j, +(ij)} x_{ij} + \sum_{i < j, -(ij)} (1 - x_{ij}) \geq \sum_{i < j, +(ij)} (x_{uj} - x_{ui}) + \sum_{i < j, -(ij)} (1 - x_{uj} - x_{ui})$$

By writing p_j for the number of '+' edges (i, j) with $i < j$, and n_j for the number of '-' edges (i, j) with $i < j$, we can rewrite this as

$$p_j x_{uj} + n_j (1 - x_{uj}) - \sum_{i < j} x_{ui}$$

Because the algorithm chose not to have a singleton cluster, the average distance of all nodes to u is at most $\frac{1}{4}$. The last sum $\sum_{i < j} x_{ui}$ only leaves out some subset of nodes furthest away from u , so the average distance of those nodes is also at most $\frac{1}{4}$. Hence, the last sum is at most $\frac{1}{4}(p_j + n_j)$, and the entire LP cost is at least

$$p_j x_{uj} + n_j (1 - x_{uj}) - \frac{p_j + n_j}{4} = p_j (x_{uj} - \frac{1}{4}) + n_j (1 - x_{uj} - \frac{1}{4}).$$

The number of negative edge mistakes the algorithm will make is at most n_j . On the other hand, because $\frac{3}{8} \leq x_{uj} \leq \frac{1}{2}$, the LP cost is at least $\frac{p_j}{8} + \frac{n_j}{4}$. Thus, the sum of edge costs incurred by the algorithm is at most four times that of the LP. Since this holds for any cluster, and we only account for the LP cost of any edge once, we have shown that the algorithm is a 4-approximation for '-' edges. ■

References

- [1] N. Bansal, A. Blum, and S. Chawla *Correlation clustering*. Machine Learning 56:89-113, 2004.
- [2] M. Charikar, V. Guruswami, and A. Wirth *Clustering with qualitative information*. Proc. 44th IEEE FOCS (2003).