Combinatorial Correlation Clustering on general unweighted graphs

Géraud Le Falher

January 7, 2015

1 Problem

In CORRELATION CLUSTERING, we have a set of n objects and a binary relationship denoting similarity $(+)$ or dissimilarity $(-)$ between these objects. Such data are naturally associated with a signed graph $G = (V, E)$. The objective is to cluster them according to this similarity information.

In this clustering problem, the number of clusters is part of the solution as opposed of being an input constraint, which is sometime desirable in unsupervised setting. Furthermore, it does not require a distance between points (although such extra information can be taken into account through edges weight).

Given any node partition, a *disagreements edge* is a positive edge within a cluster or a negative edge between different clusters. A natural complexity measure for the edge labeling (the signs associated with edges) is the minimum number of disagreement edges over **all** possible node partitions of the node set. The number of disagreements created by a given clustering is called its cost.

The problem of finding a partition minimizing the number of disagreements edges was showed to be NP-hard, even on complete graph (Bansal *et al.* 2002). Yetthere exists an elegant algorithm by Ailon *et al.* (2008) (referred later as CC-PIVOT) that runs in $O(n\,|\mathcal{P}_{\text{CC-PIVOT}}|)$, where $|\mathcal{P}_{\text{CC-PIVOT}}| \leq |V|$ is the number of clusters found with this strategy. It is a randomized algorithm with an expected approximation ratio of 3.

When the input is a general graph, the problem becomes APX-hard (Charikar *et al.* 2003; Demaine *et al.* 2006), and these two papers provide polynomial algorithms achieving $O(\log n)$ approximation. Yet they resort to expensive semi definite programs so we are seeking a combinatorial approach that would transform a graph into a clique, in order to apply the CC-PIVOT algorithm.

2 Method

A *triangle* is any set of three distinct nodes, regardless of their order. One triangle is *closeable* if it satisfies the following conditions:

1. it has exactly two edges
2. at least one edge is positive

Such triangles have a special node called *center* which is the endpoint of the two edges (see Figure 1).

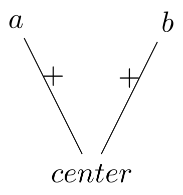

Figure 1: A closeable triangle

We obtain a clique by iteratively closing these triangles; by adding to them a third edge whose sign is the product of the signs of the two existing edges.

The crucial decision is how to choose the triangles to close during the current iteration. For that, we define a strategy as a pair of boolean (`pivot`, `one_at_a_time`). If `pivot` is true, we first select a vertex p uniformly at random from V and we restrict candidate triangles to have p as center. Otherwise, we consider all possible closeable triangles as candidates. Then, if `one_at_a_time` is true, we select one triangle uniformly at random from the candidates, otherwise we select all of them. These four strategies are named in Table 1 and the procedure just discussed is formally described below as PickTriangles, where C is the set of all currently closeable triangles in G.

```
function PickTriangles($G = (V, E)$, $strategy$, $C$)
    if $strategy$.pivot then
        $pivot \leftarrow$ one element from $V$ selected uniformly at random
        $candidates \leftarrow \{t \in C : center(t) = pivot\}$
    else
        $candidates \leftarrow C$
    if $strategy$.one_at_a_time then
        return a element of $candidates$ uniformly at random
    else
        return $candidates$
```

The full algorithm is presented in Algorithm 1. Its input is a graph G with node set $V = \{0, 1, \ldots, n-1\}$ and edge set E.

3 Experiment

We try our implementation on different controlled geometries to see how it behaves. Most of them involve *bad cycle*, that is cycle with exactly one negative edge that will necessarily

Table 1: Short names of the different randomization strategy

		pivot	
		True	False
one_at_a_time	True	P_1	N_1
	False	P_*	N_*

```
Algorithm Complete($G = (V, E)$, strategy)
  Note: in the following, triangle $= (a, center, b)$, as shown in Figure 1
  $C \leftarrow \emptyset$
  for all triplets of distinct nodes $(a, v, b)$ in $V^3$ do
      if $(a, v, b)$ is a closeable triangle then
          $C \leftarrow C \cup \{(a, v, b)\}$
  while $C \neq \emptyset$ do
      $just\_closed \leftarrow \emptyset$
      for all triangle in PickTriangles($C$) do
          Close(triangle)
          $just\_closed \leftarrow just\_closed \cup \{triangle\}$
      for all triangle in just_closed do
          Update($C$, triangle)
  set remaining edges to negative

  function Close(triangle)
      add edge $(a, b)$
      $sign(a, b) \leftarrow sign(a, center) \times sign(center, b)$

  function Update($C$, triangle $= (a, center, b)$)
      $N_a \leftarrow$ neighbors of $a$
      $N_b \leftarrow$ neighbors of $b$
      for all $v \in (N_a \cap N_b) \setminus \{a, b\}$ do
          $C \leftarrow C \setminus \{(a, v, b)\}$
      for all $v \in N_a$ do
          if $(v, a, b)$ is closeable then
              $C \leftarrow C \cup \{(v, a, b)\}$
      for all $v \in N_b$ do
          if $(v, b, a)$ is closeable then
              $C \leftarrow C \cup \{(v, b, a)\}$
```

incurs at least one disagreement. Because our algorithm is randomized, we complete each graph 50 times. On each completed graph, we run the CC-Pivot algorithm 100 times, to obtain an estimation of the expected cost of the complete graph. In all the following tables, we report the number of disagreement edges among edges of the original graph.

3.1 One bad cycle of length n

The optimal solution is either one or two connected clusters, yielding one disagreement. Strategy P_* achieves this optimum while P_1 number of errors grows linearly with the length n of the cycle. The same is true when we do not select pivot. (the missing number are due to lack of time).

n	8	16	32	64	128	256	384	512	1024
P_1	1.4	3.1	7.3	17.2	40.0	86.2	132.7	179.6	371.0
P_*	1.0	1.0	1.0	1.0	1.0	1.0	–	–	–
N_1	1.4	2.6	5.1	7.3	11.2	17.9	–	–	–
N_*	1.0	1.0	1.0	1.0	1.0	1.0	–	–	–

3.2 Cycle of length 100 having only two negative edges separated by k vertices

Because there is more than one negative edge, the optimal solution cost is 0. It is found consistently by both strategies.

k	0	1	25	50
P_1	0.0	0.0	0.0	0.0
P_*	0.0	0.0	0.0	0.0

3.3 Bad squares sharing only one (positive) edge

Here, the optimal solution is to create two clusters in the following ways. Say the shared edge is between vertices 0 and 1. Each of these two vertices is put in one cluster. Starting from 0, we can follow a path on every cycle and put vertices we cross in 0's cluster, until we encounter the negative edge of the cycle. We do the same with 1. The only disagreement is the positive edge between 0 and 1, therefore the optimal cost is 1.

number of squares	40	65	80	130
P_1	2.9	2.6	2.9	2.9
P_*	24.1	42.8	53.4	76.2
N_1	49.9	84.8	108.6	184.2
N_*	40.0	65.0	80.0	130.0

Unfortunately, all but the P_1 strategy fail to achieve this solution, instead committing a number of errors linear with the number of bad cycles.

3.4 Bad squares sharing only the (negative) edge

If we move the position of the negative edge to the shared edge, the results do not change.

number of squares	40	65	80	130	170
P_1	2.0	1.9	2.1	4.6	2.0
P_*	21.8	31.7	42.6	62.6	88.2

3.5 $\lfloor\sqrt{n}\rfloor$ bad cycles of length $\lfloor\sqrt{n}\rfloor$ sharing one positive edge

Since the length of the cycle does not affect the optimal solution, its cost is still 1. But this case is challenging for all strategies.

cycle length	7	9	12	15	20
P_1	10.6	14.7	16.3	32.5	55.1
P_*	6.8	8.8	12.0	15.1	20.1
N_1	17.2	23.9	32.2	41.6	58.7
N_*	9.1	12.1	19.8	19.3	35.3

3.6 Planted clustering

We create k clusters $\{C_1, \dots, C_k\}$ of roughly the same size n_i with around $2n$ positive edges inside them and $n_i \cdot n_j$ negative edges across each others. Then we flip a fraction $p = 0.07$ of edges at random. Since p is "small", this give us an estimation of the optimal number of disagreements, provided that the clustering stays the same. A similar model is considered by Makarychev *et al.* (2014).

In the table below, we report the number of disagreements divided by the number of flipped edges (because being random, all the graphs do not have the same number of edges, making comparison unfair). If the original clustering was not too much perturbed, this is an estimation of the approximation ratio of our method.

k	5	10	30	20	15	2	2	2	2
n	15	25	6	12	35	20	40	65	100
nodes	75	250	180	240	525	40	80	130	200
P_*	2.2	2.1	1.6	1.7	1.9	2.0	2.5	2.6	3.0
P_1	3.2	2.8	1.7	2.0	2.4	4.5	5.9	6.5	7.0

4 Discussion

None of the strategy work well in all case. P_1 is good for many squares whereas P_* is optimal for one long bad cycle, and both fail in the middle case of subsection 3.5.

Yet by providing a little help to the algorithm, we can greatly reduce the number of mistakes. Namely, if we avoid selecting as pivot the endpoints of the shared edge during the first $n \log n$ iterations (or some other arbitrary threshold), the number of disagreements is vastly reduced, for instance, it drops to 1.2 for $n = 12 \times 12$ in the case of subsection 3.5. One way to get closer to this behavior is to discard sampling uniformly at random. This can be done in many ways, here are three of them:

Preferential attachment We maintain for each node p a count p_c of how many time p has been selected as pivot so far. Then at each iteration, the pivot is chosen proportionally to $1 + p_c^\alpha$ for some parameter α. In the beginning, each node is roughly selected as often as the others, like in P_1. But at some point, a few become preeminent and their triangles are closed at a much higher rate, thus mimicking P_*.

Degree sequence This one is a modification of P_*. From the original graph G, we create a vector $D = \{d_1, \dots, d_k\}$ containing the unique elements of the sorted degree sequence of G. That is, d_1 is the smallest degree, and $d_2 > d_1$ is the next larger degree. We set $i = 1$ initially and in PickTriangles, we choose pivot only among nodes with degree less or equal than d_i. When there is no more possible choice, we increment i. In the long bad cycle case, all nodes have degree 2 so this is exactly P_*, which is optimal. When many cycles share an edge, its endpoints have high degree and are thus selected only at the end.

Note that although it should work well in these two situations, we may devise graphs for which it is not the case. Consider an input graph formed by generating $\Theta(\sqrt{n})$ bad squares sharing the same edge (i, j) and adding, for each node k (different from i and j) belonging to a bad square, $\Theta(\sqrt{n})$ new nodes having degree 1 and adjacent only to node k, such that a) $|V| = \Theta(n)$ and b) the degree of each nodes belonging to a bad square is the same: $\Theta(\sqrt{n})$.

For this input graph, this strategy turns out to be similar to P_*. Thus, according to our experiments on $\Theta(\sqrt{n})$ bad squares sharing one edge, we conjecture to have a number of disagreements linear in the number of bad cycles sharing the same edge, i.e. $\Omega(\sqrt{n})$. However, if the input graph as a (expected) bounded degree (e.g. preferential attachment models), then this strategy should achieve good performance in practice.

Phase The previous strategy strategy can be generalized by aggregating node in different *phases* that are not based on the degree of the nodes in the original input graph.

In the phase 1, we are allowed to close triangles having two edges belonging to the original input graph. In each phase $K > 1$ we are allowed to close triangles with two edges belonging to the current graph built using all phases $1, 2, \dots, K - 1$. The final constraint is that we can start each phase K only after all closable triangles in phase $K - 1$ had been closed.

Other unsorted comments:

- Another way of experimenting, leaning more toward link classification, is to take a subgraph of signed network (like Slashdot or Epinions (Leskovec *et al.* 2010)), hide some edges, cluster nodes with our algorithm and predict signs according to whether edge endpoints are in the same cluster or not.

- Once we choose one strategy, do analysis in complement to experiments
- The current algorithm has $O(n^3)$ time and space complexity and thus is not very scalable. Furthermore, because we are looking for a complete graph, it is not immediately clear how to take advantage of a distributed setting.

References

[1] N. Ailon, M. Charikar, and A. Newman, “Aggregating inconsistent information”, *Journal of the ACM*, vol. 55, no. 5, pp. 1–27, 2008.

[2] N. Bansal, A. Blum, and S. Chawla, “Correlation clustering”, *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 238–247, 2002.

[3] M. Charikar, V. Guruswami, and A. Wirth, “Clustering with qualitative information”, in *44th Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 524–533.

[4] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, “Correlation clustering in general weighted graphs”, *Theoretical Computer Science*, vol. 361, no. 2-3, pp. 172–187, 2006.

[5] J. Leskovec, D. Huttenlocher, and J. Kleinberg, “Predicting positive and negative links in online social networks”, in *Proceedings of the 19th international conference on World wide web - WWW ’10*, 2010, p. 641.

[6] K. Makarychev, Y. Makarychev, and A. Vijayaraghavan, “Algorithms for Semi-random Correlation Clustering”, 2014.