Semi-supervised clustering for de-duplication

Shrinu Kushagra
University of Waterloo

Shai Ben-David
University of Waterloo

Ihab F. Ilyas
University of Waterloo

Abstract

Data de-duplication is the task of detecting multiple records that correspond to the same real-world entity in a database. In this work, we view de-duplication as a clustering problem where the goal is to put records corresponding to the same physical entity in the same cluster and putting records corresponding to different physical entities into different clusters.

We introduce a framework which we call promise correlation clustering. Given a complete graph G with the edges labeled 0 and 1, the goal is to find a clustering that minimizes the number of 0 edges within a cluster plus the number of 1 edges across different clusters (or correlation loss). The optimal clustering can also be viewed as a complete graph G^* with edges corresponding to points in the same cluster being labeled 0 and other edges being labeled 1. Under the promise that the edge difference between G and G^* is "small", we prove that finding the optimal clustering (or G^*) is still NP-Hard. [Ashtiani et al., 2016] introduced the framework of semi-supervised clustering, where the learning algorithm has access to an oracle, which answers whether two points belong to the same or different clusters. We further prove that even with access to a same-cluster oracle, the promise version is NP-Hard as long as the number queries to the oracle is not too large ($o(n)$ where n is the number of vertices).

Given these negative results, we consider a restricted version of correlation clustering. As before, the goal is to find a clustering that minimizes the correlation loss. However, we restrict ourselves to a given class $\mathcal{F}$ of clusterings. We offer a semi-supervised algorithmic approach to solve the restricted variant with success guarantees.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

1 Introduction

Record de-duplication is a central task in data cleaning in large data bases. Common practical examples include the detection of records referring to the same patient in large health data bases (different records might have been generated for same patient in different clinics or even in the same clinic at different times), detecting same person records in census data, detecting customer records, duplicate records of papers in Google Scholar and so on and so forth [Elmagarmid et al., 2007], [Chu et al., 2016], [Ilyas et al., 2015].

Since the same-entity relation is reflexive, symmetric and transitive, the sets of duplicate records can be viewed as clusters. Consequently, the record de-duplication task can be viewed as a clustering task. Such a clustering task has several characteristics that make it hard to address with common clustering tools; The number of ground truth clusters is unknown to the algorithm. Furthermore, one cannot a priory bias the algorithm towards a larger or a smaller number of clusters (unlike, say, facility location tasks in which it makes sense to trade off cluster cohesiveness with the number of clusters). This implies that attempts to use standard classification prediction learning tools to predict which pairs or records should be labelled 'same-cluster' and which should be 'different clusters' (D) are bound to fail - uniformly drawn samples of pairs are likely to be all labeled D and the resulting constant "all D" classifier will have negligible $0-1$ error over the set of pairs. On top of that all, there is no a priori geometry to the clusters structure, one cannot justify common simplifying assumptions like some stability of the clustering, some convexity of the larger-than-two sized clusters or significant between-clusters margins.

The framework of correlation clustering extends naturally to the data de-duplication problem [Bansal et al., 2004]. Given a complete graph G where each edge is labelled as a 1 or a 0, the goal is to cluster the vertices of the graph so as to correlate "as much as possible" to the edges of the graph. That is, find a clustering so as to minimize the number of 0 edges within a cluster plus the number of 1 edges across different clusters (or correlation loss). An edge label 0 indicates that the two records have been deemed to be different while 1 indicates that the records are similar. However, finding the clustering with minimum correlation loss is known to be NP-Hard [Bansal et al., 2004].

One characteristic of record de-duplication which makes it different from other clustering tasks is the applicability of 'human supervision'. For example, given two records from a medical database or two papers from DBLP or two citizens from a census data, it is fairly easy for a human to identify whether these records refer to the same physical entity. The framework of correlation clustering does not take this into account. Most prevalent approaches for data de-duplication are based on designing a similarity measure (or distance) over the records, such that records that are highly similar according to that measure are likely to be duplicate and records that measure as significantly dissimilar are likely to represent different entities. In other words, the edge labels are 'close to' the underlying ground truth clustering. This is another aspect of data de-duplication which the current correlation clustering framework does not take into account.

In this paper we offer a formal modeling of such record de-duplication tasks. Our framework is the same as correlation clustering but with the added *promise* that the input graph edges E is 'close to' the optimal correlation clustering of the given dataset. We analyse the computational complexity of this problem and show that even under strong promise, correlation clustering is NP-Hard. Moreover, the problem remains NP-Hard (assuming the ETH hypothesis) even when we are allowed to make queries to a human expert (or an *oracle*) as long as the number of queries is not too large (less than the number of points in the dataset).

Given these negative results, we propose a *restricted* variant of correlation clustering. Here, instead of finding the best clustering from the class of all possible clusterings, the learning algorithm has to choose the best clustering from a given class $\mathcal{F}$ of clusterings. We offer an algorithmic approach (which uses the help of an oracle) with success guarantees for the restricted version. The 'success guarantee' depends on the complexity of the class $\mathcal{F}$ (measured by VC-Dim($\mathcal{F}$)) as well as the 'closeness' of the metric d to the target clustering.

1.1 Related Work

The most relevant work is the framework of correlation clustering developed by [Bansal et al., 2004] that we discussed in the previous section. Other variations of correlation clustering have been considered. For example [Demaine et al., 2006], consider a problem where the edges can be labelled by a real number instead of just 0 or 1. Edges with large positive weights encourage those vertices to be in the same cluster while edges with large negative weights encourage those points to be in different clusters. They showed that the problem is NP-Hard and gave a $O(\log n)$ approximation to the weighted correlation clustering problem. [Charikar et al., 2005] made several contributions to the correlation clustering problem. For the problem of minimizing the correlation clustering loss (for unweighted complete graphs), they gave an algorithm with factor 4 approximation. They also proved that the minimization problem is APX-Hard.

More recently, [Ailon et al., 2018] considered the problem of correlation clustering in the presence of an oracle. If the number of clusters k is known, they proposed an algorithm which makes $O(k^{14}\log n)$ queries to the oracle and finds a $(1+\epsilon)$-approximation to the correlation clustering problem. They showed that the problem is NP-Hard to approximate with $o\big(\frac{k}{poly\log k}\big)$ queries to an oracle. In this work, we obtain similar results for the promise correlation clustering problem.

Supervision in clustering has been addressed before. For example, [Kulis et al., 2009, Basu et al., 2004, Basu et al., 2002] considered *link/don't-link* constraints. This is a form of non-interactive clustering where the algorithm gets as input a list of pairs which should be in the same cluster and a list pairs which should be in different clusters. [Balcan and Blum, 2008] developed a framework of interactive clustering where the supervision is provided in the form of *split/merge* queries. The algorithm gives the current clustering to the oracle. The oracle responds by telling the which clusters to merge and which clusters to split.

In this work, we use the framework of same-cluster queries developed by [Ashtiani et al., 2016]. At any given instant, the clustering algorithm asks the same-cluster oracle about two points in the dataset. The oracle replies by answering either 'yes' or 'no' depending upon whether the two points lie in the same or different clusters.

On de-duplication side, most prevalent are approaches that are based on designing a similarity measure (or distance) over the records, such that records that are highly similar according to that measure are likely to be duplicates and records that measure as significantly dissimilar are likely to represent different entities. For example, to handle duplicate records created due typographical mistakes, many character-based similarity metrics have been considered. Examples of such metrics include the edit or levenshtein distance [Levenshtein, 1966], smith-waterman distance [Smith and Waterman, 1981] and jaro distance metric [Jaro, 1980]. Token-based similarity metrics try to handle rearrangement of words, for example [Monge et al., 1996] and [Cohen, 1998]. Other techniques include phonetic-based metrics and numerical metrics (to handle numeric data). A nice overview of these methods can be found in [Elmagarmid et al., 2007].

While the above approaches relied on designing a good similarity metric, some works try to 'learn' the distance function from a labelled training dataset of pairs of records. Examples of such works include [Cochinwala et al., 2001] and [Bilenko et al., 2003]. Clustering for de-duplication has been mostly addressed in application oriented works. [Hernández and Stolfo, 1995] assumes that the duplicate records are transitive. The clustering problem now reduces to finding the connected components in a graph.

1.2 Outline

Section 2 introduces the relevant notation and definitions. In Section 3, we introduce our framework of Promise Correlation Clustering. In Section 3.1, we prove that PCC is NP-Hard. In Section 3.2 we prove that PCC is NP-Hard even under the presence of an oracle. In Section 4, we introduce our framework of Restricted Correlation Clustering (RCC). In Sections 4.1 and 4.2 we describe procedures for sampling different-cluster (negative) and same-cluster (positive) pairs. In Section 5, we describe our semi-supervised algorithm for solving the RCC problem. We prove an upper bound on the number of labelled samples required to guarantee the success of our algorithm. We also upper bound the number of queries made to the same-cluster oracle. Section 6 concludes our work. All the missing proofs can be found in the supplementary section.

2 Preliminaries

Given a finite domain X. A clustering C of the set X is a partition of the set X into k disjoint subsets, that is, $C = \{C_1, \dots, C_k\}$. Denote by $m(C) = \max C_i$. Define $X^{[2]} = \{(x, y) : x \neq y\}$. In this paper, we view a clustering as a binary-valued function over the pairs of instances. That is, $C : X^{[2]} \to \{0, 1\}$ and $C(x, y) = 1$ if and only if x, y are in the same C cluster.

Given $G = (X, E)$, define $d_E(x, y) = 0$ if there exists an edge between x, y and $d_E(x, y) = 1$ otherwise.

Definition 1 (Correlation clustering for deduplication)**.** *[Bansal et al., 2004] Given $G = (X, E)$, find a clustering C which minimizes*

$$\begin{aligned} &L_{d_E}(C) = NL_{d_E}(C) + PL_{d_E}(C), \text{ where} \\ &NL_{d_E}(C) = |\{(x, y) : C(x, y) = 1 \text{ and } d_E(x, y) = 0\}|, \\ &PL_{d_E}(C) = |\{(x, y) : C(x, y) = 0 \text{ and } d_E(x, y) = 1\}| \end{aligned} \tag{1}$$

$L_{d_E}(C)$ is also referred to as the correlation loss. A weighted version of the loss function places weights of w_1 and w_2 on the two terms and is defined as

$$L^{w_1,w_2}_{d_E}(C) = w_1 NL_{d_E}(C) + w_2 PL_{d_E}(C) \tag{2}$$

Definition 2 (Informative metric)**.** *Given (X, d), a clustering C^* and a parameter λ. We say that the metric d is (α, β)-informative w.r.t C^* and λ if*

$$\underset{(x,y)\sim U^2}{\mathbf{P}} \left[d(x, y) > \lambda \mid C^*(x, y) = 1\right] \leq \alpha \tag{3}$$

$$\underset{(x,y)\sim U^2}{\mathbf{P}} \left[C^*(x, y) = 1 \mid d(x, y) \leq \lambda\right] \geq \beta \tag{4}$$

Here U^2 is the uniform distribution over $X^{[2]}$.

This definition says that most of the same-cluster (or positive) pairs are such that the distance between them is atmost λ. Also, atleast a β fraction of all pairs with distance $\leq \lambda$ belong to the same cluster.

To incorporate supervision into the clustering problem, we allow an algorithm to make *same-cluster* queries to a C^*-oracle defined below.

Definition 3 (Same-cluster oracle [Ashtiani et al., 2016])**.** *Given X. A same-cluster C^*-oracle receives a pair $x, y \in X$ as input and outputs 1 if x, y belong to the same-cluster according to C^*. Otherwise, it outputs 0.*

In the next section, we introduce our framework of *promise correlation clustering* and discuss the computational complexity of the problem both in the absence and presence of an oracle.

3 Promise Correlation Clustering

Definition 4 (Promise correlation clustering (PCC))**.** *Given a clustering instance $G = (X, E)$. Let C^* be such that*

$$C^* = \underset{C\in\mathcal{F}}{\arg\min}\ L_{d_E}(C) \tag{5}$$

where $\mathcal{F}$ is the set of all possible clusterings C such that $m(C) \leq M$. Given that d_E is (α, β)-informative. Find the clustering C^*.

When the edges E correspond to a clustering C then $\beta = 1$ and $\alpha = 0$. We show in the subsequent sections that even when the size of the maximum cluster is atmost a constant M and given the prior knowledge, PCC is still NP-Hard. Furthermore, PCC is NP-Hard even when we are allowed to make $o(|X|)$ queries to a C^*-oracle.

3.1 PCC is NP-Hard

Theorem 5. *Finding the optimal solution to the Promise Correlation Clustering problem is NP-Hard for all $M \geq 3$ and for $\alpha = 0$ and $\beta = \frac{1}{2}$.*

To prove the result, we will use a reduction from exact cover by 3-sets problem which is known to be NP-Hard.

(X3C) Given a universe of elements $U = \{x_1, \ldots, x_{3q}\}$ and a collections of subsets $S = \{S_1, \ldots, S_m\}$. Each $S_i \subset U$ and contains exactly three elements. Does there exist $S' \subseteq S$ such that each element of U occurs exactly once in S'?

This decision problem is known to be NP-Hard. We will now reduce an instance of X3C to the promise correlation clustering problem. For each three set $S_i = \{x_{i1}, x_{i2}, x_{i3}\}$, we construct a replacement gadget as described in Fig. 1. The gadget is similar to the one used in the proof of partition into triangles problem. However, instead of triangles the graph is 'made of' cliques of size M.

Given an instance of X3C, we construct $G = (V, E)$ using local replacement described in Fig. 1. Let A be an algorithm which solves the promise problem described in Eqn. 5. Then, we can use this algorithm to decide exact cover by three sets as follows.

If A outputs a clustering C such that all the clusters have size exactly M and E_C makes no negative errors w.r.t E (that is $\alpha(E_C) = 0$) then output YES. Otherwise, output NO. Next, we will prove that this procedure decides X3C.

Let there exists an exact cover for the X3C instance. Let C be the clustering corresponding to the exact cover. That is, the edges colored blue and black correspond to this clustering and the corresponding vertices are in the same cluster (Fig. 1). Note that this clustering makes no negative errors. Furthermore, each point is in a cluster of size exactly M. Thus, the positive error corresponding to any vertex is the degree of that vertex minus $M - 1$. Since, the size of a cluster is atmost M, this is the minimum possible positive error for any vertex. Hence, any other clustering strictly makes more positive errors than C.

It is easy to see from the construction that if A finds a clustering which has no negative errors and all the clusters have size M, then this corresponds to exact cover of the X3C instance and hence we output YES. If this does not happen then there does not exist any exact cover for (U, S). This is because if there was an exact cover then the corresponding clustering would satisfy our condition. Thus, A decides X3C. Since, X3C is NP-Hard, no polynomial time algorithm A exists unless $P = NP$.

In the construction, for each clause, we have $M^2 t + (M-3)$ vertices and a vertex for each of the variables. Therefore, $|V| = m(M^2 t + (M-3)) + 3q$ and $|E| = Mt(\binom{M}{2} + M - 1) + \binom{M}{2}$. Consider a clustering C which places all the x_i's and r_i's in singleton clusters and places rest of the points in clusters of size M. For $t \geq 2$,

$$\beta = \frac{Mt\binom{M}{2}}{Mt(\binom{M}{2} + M - 1) + \binom{M}{2}} = \frac{1}{1 + \frac{2}{M} + \frac{1}{Mt}} > \frac{1}{2}$$

and $\alpha = 0$

3.2 Hardness of PCC in the presence of an oracle

In the previous sections, we have shown that the PCC problem is NP-Hard without queries. It is trivial to see that by making $\beta|X|$ queries to the same-cluster oracle allows us to solve (in polynomial time) the Promise Correlation Clustering problem for all M and $\alpha = 0$. In this section, we prove that the linear dependence on $n = |X|$ is tight. We prove that if the exponential time hypothesis (ETH) holds then any algorithm that runs in polynomial time makes atleast $\Omega(n)$ same-cluster queries.

Theorem 6. *Given that the Exponential Time Hypothesis (ETH) holds then any algorithm for the Promise Correlation Clustering problem that runs in polynomial time makes $\Omega(|X|)$ same-cluster queries for all $M \geq 3$ and for $\alpha = 0$ and $\beta = \frac{1}{2}$.*

Below, we give a proof sketch but a detailed proof is in the supplementary material. The exponential time hypothesis says that any solver for 3-SAT runs in $2^{o(m)}$ time (where m is the number of clauses in the 3-SAT formula). We use a reduction from 3-SAT to 3DM to X3C to show that the exact cover by 3-sets (X3C) problem also can't be solved in $2^{o(m)}$ time (if ETH holds). Then, using the reduction from the previous section implies that PCC also can't be solved in $2^{o(n)}$ time. Thus, any query based algorithm for PCC needs to make atleast $\Omega(n)$ queries where $n = |X|$ is the number of vertices in the graph.

Definition 7 (3-SAT)**.** .
Input: A boolean formulae ϕ in 3CNF with n literals

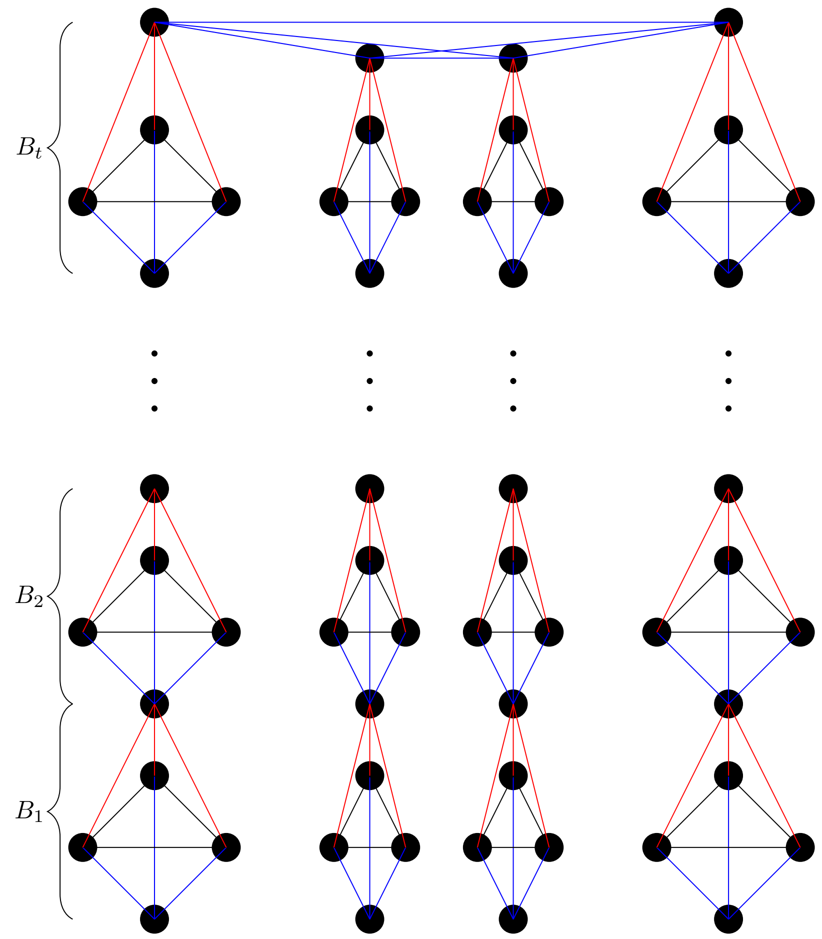

Figure 1: Part of graph G constructed for the subset $S_i = \{x_{i1}, x_{i2}, x_{i3}\}$. The graph is constructed by local replacement when for $p = 4$. If S_i is included in the exact cover then the edges colored black and the edges colored blue represent the corresponding clustering of this part of the graph G. If S_i is not included in the exact cover then the edges colored red and the edges colored black represent the clustering of this part of the graph.

and m clauses. Each clause has exactly three literals. Output: YES if ϕ is satisfiable, NO otherwise.

Exponential Time Hypothesis
There does not exist an algorithm which decides 3-SAT and runs in $2^{o(m)}$ time.

To prove that (X3C) is NP-Hard, the standard We will reduce 3-SAT to 3-dimensional matching problem. 3DM is already known to be NP-Hard. However, the standard reduction of 3-SAT to 3DM constructs a set with number of matchings in $\Theta(m^2n^2)$. Hence, using the standard reduction, the exponential time hypothesis would imply there does not exist an algorithm for 3DM which runs in $\Omega(m^{\frac{1}{4}})$. Our reduction is based on the standard reduction. However, we make some clever optimizations especially in the way we encode the clauses. This improves the lower bound to $\Omega(m)$.

Using the above result, we immediately get an $\Omega(2^m)$ lower bound on the run-time of X3C. Now, using the same reduction of X3C to PCC as in Section 3.1, gives the same lower bound of $\Omega(n)$ on the running time of PCC.

For the sake of contradiction, let us assume that there exists an algorithm which solves PCC in polynomial time by making $o(n)$ same-cluster queries (n is the number of vertices). Then by simulating all possible answers for the oracle, we get a non-query algorithm which solves PCC in $2^{o(n)}$. Hence, no such query algorithm exists.

4 Restricted Correlation Clustering

The results in the previous section show that even under strong promise, correlation clustering is still NP-

Hard. Furthermore, it is hard even when given access to an oracle. This motivates us to consider a restricted version of the problem.

Note that in Defn. 4, the optimization problem was over the set of all possible clusterings $\mathcal{F}$ (with a restriction on the maximum cluster size). In this section, we restrict $\mathcal{F}$ to be a finite class of clusterings. That is, $\mathcal{F}' = \{T_1, \ldots, T_r, C_1, \ldots, C_s\}$ with the understanding that each T_i (a hierarchical clustering tree of X) is a collection of clusterings represented by the prunings of the tree. Now, we consider two versions of correlation clustering on this restricted family. The first is to find a clustering $C \in \mathcal{F}'$ which correlates 'as much as possible' with the given graph $G = (X, E)$. More formally, given $G = (X, E)$ find $\hat{C} \in \mathcal{F}'$ such that

$$\hat{C} = \arg\min_{C \in \mathcal{F}'} \; L_E(C) \tag{6}$$

Eqn. 6 can be solved by going over the list of clusterings and trees (in a bottom-up fashion) and in polynomial time finding $\hat{C}$ which is 'closest' to E. In the second version, the goal is to find a clustering $\hat{C} \in \mathcal{F}'$ which correlates as much as possible to an unknown target clustering C^* which may or may not be in the set $\mathcal{F}'$. However, the algorithm has access to a C^*-oracle. For the rest of this paper, we will focus on the second version which we call *restricted correlation clustering.*

Definition 8 (Restricted correlation clustering (RCC))**.** *Given a clustering instance (X, d). Let C^* be an unknown target clustering of X and weights w_1, w_2. Let $d_{C^*} : X^{[2]} \to \{0, 1\}$ be defined as $d_{C^*}(x, y) = 0$ if x, y are in the same C^* cluster and 1 otherwise. Find $\hat{C} \in \mathcal{F}'$ such that*

$$\hat{C} = \arg\min_{C \in \mathcal{F}'} \; L^{w_1, w_2}_{d_{C^*}}(C) \tag{7}$$

where $\mathcal{F}' = \{T_1, \ldots, T_r, C_1, \ldots, C_s\}$. T_i is a hierarchical clustering tree and C_i is a clustering of X.

To solve the RCC problem, we adopt the following strategy. We use a procedure (call it $\mathcal{P}_0$) to sample negative (or different cluster) pairs and another procedure (call it $\mathcal{P}_1$) to sample positive (or same-cluster) pairs. Both the sampling procedures use the help of the C^*-oracle. We then evaluate each of the clusterings in $\mathcal{F}'$ on our sample S and choose the clustering which has minimum loss. We prove that the loss of the clustering $\hat{C}$ obtained using this procedure is close to the loss of $\hat{C}^*$ (the clustering with minimum loss in $\mathcal{F}'$) .

We first discuss how to sample the positive and negative pairs. Then, we discuss the sample complexity of our approach. That is, the number of positive and negative pairs (or $|S|$) needed to guarantee that the loss of $\hat{C}$ is close to that of $\hat{C}^*$. Before we proceed, lets introduce the following definitions which will be useful in the subsequent sections.

Definition 9 (Restricted distributions)**.** *Given X and a target clustering C^*. Define $X^{[2]+} = \{(x, y) \in X^{[2]} : C^*(x, y) = 1\}$ and $X^{[2]-} = \{(x, y) \in X^{[2]} : C^*(x, y) = 0\}$. We define P^+ as the uniform distribution over $X^{[2]+}$ and P^- as the uniform distribution over $X^{[2]-}$.*

The sampling procedure $\mathcal{P}_0$ will try to approximate P^- while $\mathcal{P}_1$ will approximate P^+.

Definition 10 (γ-skewed)**.** *Given X and a C^*-oracle. We say that X is γ-skewed w.r.t C^* if*

$$\mathop{\mathbf{P}}_{(x,y) \sim U^2} \Big[\; C^*(x, y) = 1 \Big] \;\; \leq \;\; \gamma$$

The above definition formalizes the statement that most of the pairs of points belong to different clusters.

4.1 Sampling negative pairs

Assume our input X is γ-skewed. Thus, if we choose a pair uniformly at random, then it is 'highly likely' to be a negative pair. Alg. 1 describes our sampling procedure.

Algorithm 1: Procedure $\mathcal{P}_0$ for negative pairs

```
Input: A set $X$ and a $C^*$-oracle.
Output: One pair $(x, y) \in X^{[2]}$ such that
        $C^*(x, y) = 0$
1 while TRUE do
2     Sample $(x, y)$ using $U^2$
3     if $C^*(x, y) = 0$ then
4         Output $(x, y)$
5     end
6 end
```

Lemma 11. *Given X and a C^*-oracle. The procedure $\mathcal{P}_0$ samples a pair (x, y) according to the distribution P^-.*

Proof. The probability that a negative pair is sampled during a trial is $U^2(X^{[2]-}) =: q$. Fix a negative pair (x, y) and let $U^2(x, y) = p$. Hence, the probability that the pair (x, y) is sampled $= p + (1-q)p + (1-q)^2 p + \ldots = p \sum_{i=0}^{\infty} (1-q)^i = \frac{p}{q} = \frac{U^2(x,y)}{U^2(X^{[2]-})} = P^-(x, y)$. ☐

Note that to sample one negative pair, procedure $\mathcal{P}_0$ might need to ask more than one same-cluster query. However, since our input X is γ-skewed, we 'expect' the number of 'extra' queries to be 'small'.

Lemma 12. *Given set X and a C^*-oracle. Let X be γ-skewed and Let q be the number of same-cluster queries made by $\mathcal{P}_0$ to the C^*-oracle. Then, $\mathbf{E}[q] \leq \frac{1}{1-\gamma}$.*

Proof. Let p denote the probability that a negative pair is sampled during an iteration. We know that $p \geq (1-\gamma)$. Let q be a random variable denoting the number of iterations (or trials) before a negative pair is sampled. Then, q is a geometric random variable. $\mathbf{E}[q] = \frac{1}{p} \leq \frac{1}{1-\gamma}$. □

Lemma 12 shows that for $\gamma < \frac{1}{2}$, to sample a negative pair, procedure $\mathcal{P}_0$ makes atmost two queries to the oracle in expectation. Moreover, the number of queries is tight around the mean. Note that this sampling strategy is not useful for positive pairs. This is because the fraction of positive pairs in the dataset is small. Hence, to sample a single positive pair we would need to make 'many' same-cluster queries.

4.2 Sampling positive pairs

Given a clustering instance (X, d). Assume that the metric d is (α, β)-informative w.r.t target C^* and parameter λ. This means that 'most' of the positive pairs are within distance λ. Our sampling strategy is to construct a set $K = \{(x, y) \in X^2 : d(x, y) \leq \lambda\}$ and then sample uniformly from this set. We will prove that this procedure approximates P^+. Note that constructing the set K requires $O(|X|^2)$ time. In a some situations, the metrics d has some structure which makes it "locally sensitive hashable". In such situations, we can get rid of the quadratic dependence on $|X|$ and approximate P^+ in only $|X|$ pre-processing time. Due to space constraints we include these details only in the appendix.

The sampling algorithm is described in Alg. 2. In the pre-compute stage, for all points x we construct its set of 'neighbours' (S_x). We then choose a point with probability proportional to the size of its neighbour-set and then choose the second point uniformly at random from amongst its neighbours. This guarantees that we sample uniformly from the set K.

Lemma 13. *Given set (X, d), a C^*-oracle and parameter λ. Let d be (α, β)-informative w.r.t λ and C^*. Then the sampling procedure $\mathcal{P}_1$ induces a distribution T over $X^{[2]}$ such that for any labelling function h over $X^{[2]}$ we have that*

$$\left| \underset{(x,y)\sim P^+}{\mathbf{P}} \left[h(x,y)=0\right] - \underset{(x,y)\sim T}{\mathbf{P}} \left[h(x,y)=0\right] \right| \leq 2\alpha.$$

Note that to sample one positive pair, procedure $\mathcal{P}_1$ might need to ask more than one same-cluster query.

Algorithm 2: Sampling procedure $\mathcal{P}_1$ for positive pairs (general metrics)

```
Input: A set $X$, a $C^*$-oracle and a parameter $\lambda$.
Output: One pair $(x, y) \in X^{[2]}$ such that
        $\mathcal{C}^*(x, y) = 1$
1 Pre-compute: For all $x \in X$, compute
    $S_x := \{y : d(x, y) \leq \lambda\}$.
2 while TRUE do
3     Sample $x \in X$ with probability $\propto |S_x|$.
4     Sample $y$ uniformly at random from $S_x$.
5     if $\mathcal{C}^*(x, y) = 1$ then
6         Output $(x, y)$.
7     end
8 end
```

However, since the metric d is β-informative, we 'expect' the number of 'extra' queries to be 'small'.

Lemma 14. *Given set (X, d), a C^*-oracle and a parameter λ. Let d be β-informative w.r.t λ and let q be the number of same-cluster queries made by $\mathcal{P}_1$ to the C^*-oracle. Then, $\mathbf{E}[q] \leq \frac{1}{\beta}$.*

Proof. Let p denote the probability that a positive pair is sampled during an iteration. We know that $p \geq \beta$. Let q be a random variable denoting the number of iterations (or trials) before a positive pair is sampled. Then, q is a geometric random variable. $\mathbf{E}[q] = \frac{1}{p} \leq \frac{1}{\beta}$. □

5 Sample and query complexity of RCC

In the previous section, we developed a sampling procedure for positive and negative pairs. We showed that the procedures sample according to distributions T_1 and T_2 which approximate P^- and P^+ respectively. Given a class of clusterings $\mathcal{F}$, we use our distributions T_1 and T_2 to estimate the negative and positive components of the loss for each clustering $C \in \mathcal{F}$. We then choose the clustering $\hat{C}$ with the minimum estimated loss. Using standard VC-Dimension theory, it is easy to show that the loss of the clustering $\hat{C}$ is close to the loss of best clustering in $\mathcal{F}$, as long the VC-Dimension of $\mathcal{F}$ is finite.

The loss function $L_{d_{C^*}}^{w_1,w_2}$ is the sum of the sizes of two sets. However, in this section it would be more convenient to work with bounded loss functions. Let $\gamma_0 = \underset{(x,y)\sim U^2}{\mathbf{P}} \left[\ C^*(x,y) = 1\right]$ and define $\mu = \frac{w_1\gamma_0}{w_1\gamma_0 + w_2(1-\gamma_0)}$. Then we see that minimizing Eqn. 7, is the same as minimizing

Definition 15 (Normalized correlation loss)**.**

$$L_{C^*}(C) = \mu \underset{(x,y)\sim P^+}{\mathbf{P}}\big[C(x,y)=0\big] + (1-\mu)\underset{(x,y)\sim P^-}{\mathbf{P}}\big[C(x,y)=1\big] \quad (8)$$

For the remainder of the section, we work with this formulation of the loss function. We describe this procedure in Alg. 3.

Algorithm 3: Empirical Risk Minimization

```
Input: $(X, d)$, a set of clusterings $\mathcal{F}$, a $C^*$-oracle,
       parameter $\lambda$ and sizes $m_+$ and $m_-$.
Output: $C \in \mathcal{F}$
1 Sample a sets $S_+$ and $S_-$ of sizes $m_+$ and $m_-$
    using procedures $\mathcal{P}_1$ and $\mathcal{P}_0$.
2 For every $C \in \mathcal{F}$ and define
    $\hat{E}(C) = \frac{|\{(x,y) \in S_+ : C(x,y)=0\}|}{|S_+|}$
    $\hat{G}(C) = \frac{|\{(x,y) \in S_- : C(x,y)=0\}|}{|S_-|}$
3 Define $\hat{L}(h) = \mu\hat{E}(h) + (1-\mu)\hat{G}(h)$.
4 Output $\arg\min_{C\in\mathcal{F}} \hat{L}(l_C)$
```

Theorem 16. *Given metric space (X, d), a class of clusterings $\mathcal{F}$ and a threshold parameter λ. Given $\epsilon, \delta \in (0,1)$ and a C^*-oracle. Let d be (α,β)-informative and X be γ-skewed w.r.t λ and C^*. Let $\mathcal{A}$ be the ERM-based approach as described in Alg. 3 and $\hat{C}$ be the output of $\mathcal{A}$. If*

$$m_-, m_+ \;\; \geq a\frac{\text{VC-Dim}(\mathcal{F}) + \log(\frac{2}{\delta})}{\epsilon^2} \quad (9)$$

where a is a global constant then with probability atleast $1-\delta$ (over the randomness in the sampling procedure), we have that

$$L_{C^*}(\hat{C}) \;\; \leq \;\; \min_{\mathcal{C}\in\mathcal{F}} L_{C^*}(\mathcal{C}) + 3\alpha + \epsilon$$

Next we show that to sample m_+ positive and m_- negative pairs, the number of queries made to the C^* is not too large.

Theorem 17. *[Query Complexity] Let the framework be as in Thm. 16. With probability atleast $1-\exp\big(-\frac{\nu^2 m_-}{4}\big)-\exp\big(-\frac{\nu^2 m_+}{4}\big)$ over the randomness in the sampling procedure, the number of same-cluster queries q made by $\mathcal{A}$ is*

$$q \leq (1+\nu)\bigg(\frac{m_-}{(1-\gamma)} + \frac{m_+}{\beta}\bigg)$$

5.1 VC-Dimension of some common classes of clusterings

In the previous section, we proved that the sample complexity of learning a class of clusterings $\mathcal{F}$ depends upon VC-Dim($\mathcal{F}$). Recall that $\mathcal{F}$ is the class of labellings induced by the clusterings in $\mathcal{F}$. In this section, we prove upper bounds on the VC-Dimension for some common class of clusterings.

Theorem 18. *Given a finite set $\mathcal{X}$ and a finite class $\mathcal{F} = \{C_1, \ldots, C_s\}$ of clusterings of $\mathcal{X}$.*

$$\text{VC-Dim}(\mathcal{F}) \leq g(s)$$

where $g(s)$ is the smallest integer n such that $B_{\sqrt{n}} \geq s$ where B_i is the i^{th} bell number [A000108,].

Note that $B_{\sqrt{n}} \in o(2^n)$. Thus, the VC-Dim of a list of clusterings is in $o(\log s)$. Next, we discuss another common class of clusterings, namely hierarchical clustering trees.

Definition 19 (Hierarchical clustering tree)**.** *Given a set X. A hierarchical clustering tree T is a rooted binary tree with the elements of X as the leaves.*

Every pruning of a hierarchical clustering tree is a clustering of the set X. A clustering tree contains exponentially many (in the size of $\mathcal{X}$) clusterings. Given $\mathcal{F} = \{T_1, \ldots, T_s\}$ consists of s different hierarchical clustering trees, the following theorem bounds the VC-Dimension of $\mathcal{F}$.

Theorem 20. *Given a finite set $\mathcal{X}$ and a finite class $\mathcal{F} = \{T_1, \ldots, T_s\}$ where each T_i is a hierarchical clustering over $\mathcal{X}$. Then*

$$\text{VC-Dim}(\mathcal{F}) \leq g(s)$$

where $g(s)$ is the smallest integer n such that $\frac{\sqrt{n}!}{\lfloor\sqrt{n}/2\rfloor! \;\; 2^{\lfloor\sqrt{n}/2\rfloor}} \geq s$

6 Conclusion

We introduced a promise version of correlation clustering. We proved that the promise version is NP-Hard. Furthermore, the problem is NP-Hard even when we are allowed to make $o(|X|)$ queries to a same-cluster oracle (where X is the clustering instance). We then introduced a restricted version of correlation clustering. We developed a sampling procedure (with the help of the same-cluster oracle) to sample same-cluster and different-cluster pairs. We then used this procedure to solve the restricted variant.

References

[A000108,] A000108, S. The on-line encyclopedia of integer sequences. *published electronically at https://oeis.org, 2010.*

[Ailon et al., 2018] Ailon, N., Bhattacharya, A., and Jaiswal, R. (2018). Approximate correlation clustering using same-cluster queries. In *Latin American Symposium on Theoretical Informatics*, pages 14–27. Springer.

[Ashtiani et al., 2016] Ashtiani, H., Kushagra, S., and Ben-David, S. (2016). Clustering with same-cluster queries. In *Advances in neural information processing systems*, pages 3216–3224.

[Balcan and Blum, 2008] Balcan, M.-F. and Blum, A. (2008). Clustering with interactive feedback. In *International Conference on Algorithmic Learning Theory*, pages 316–328. Springer.

[Bansal et al., 2004] Bansal, N., Blum, A., and Chawla, S. (2004). Correlation clustering. *Machine Learning*, 56(1-3):89–113.

[Basu et al., 2002] Basu, S., Banerjee, A., and Mooney, R. (2002). Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer.

[Basu et al., 2004] Basu, S., Bilenko, M., and Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM.

[Bilenko et al., 2003] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

[Charikar et al., 2005] Charikar, M., Guruswami, V., and Wirth, A. (2005). Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383.

[Chu et al., 2016] Chu, X., Ilyas, I. F., and Koutris, P. (2016). Distributed data deduplication. *Proceedings of the VLDB Endowment*, 9(11):864–875.

[Cochinwala et al., 2001] Cochinwala, M., Kurien, V., Lalk, G., and Shasha, D. (2001). Efficient data reconciliation. *Information Sciences*, 137(1-4):1–15.

[Cohen, 1998] Cohen, W. W. (1998). Integration of heterogeneous databases without common domains using queries based on textual similarity. In *ACM SIGMOD Record*, volume 27, pages 201–212. ACM.

[Demaine et al., 2006] Demaine, E. D., Emanuel, D., Fiat, A., and Immorlica, N. (2006). Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187.

[Elmagarmid et al., 2007] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16.

[Hernández and Stolfo, 1995] Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. In *ACM Sigmod Record*, volume 24, pages 127–138. ACM.

[Ilyas et al., 2015] Ilyas, I. F., Chu, X., et al. (2015). Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, 5(4):281–393.

[Jaro, 1980] Jaro, M. A. (1980). *UNIMATCH, a Record Linkage System: Users Manual*. Bureau of the Census.

[Kulis et al., 2009] Kulis, B., Basu, S., Dhillon, I., and Mooney, R. (2009). Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[Monge et al., 1996] Monge, A. E., Elkan, C., et al. (1996). The field matching problem: Algorithms and applications. In *KDD*, pages 267–270.

[Smith and Waterman, 1981] Smith, T. and Waterman, M. (1981). Identification of common molecular subsequence. *J Mol. Biol*, 147.