1 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.

Observe that the requirement of correlation clustering is very demanding. The algorithm is required to find a clustering over the set of all possible clusterings of the domain X. In the restricted framework, we change the goalpost slightly. The algorithm is now required to find a clustering C from a class $\mathcal{F}$ (of clusterings of X).

Definition 1 (Restricted correlation clustering (RCC))**.** *Given a clustering instance (X, d), an unknown target clustering C^* and weighting parameter μ. Given a finite class $\mathcal{F}$ of clusterings of the set X. Find $C \in \mathcal{F}$ such that*

$$\hat{C} = \arg\min_{C \in \mathcal{F}} \; L_{C^*}(C) \tag{1}$$

1.1 Relation to practical applications

Consider the following scenario from the practitioner's point of view. The practitioner wants to implement correlation clustering. However, he/she knows that the problem is NP-Hard. The practitioner has prior knowledge that one of the many hierarchical clustering algorithms (like single-linkage or max-linkage or average-linkage or complete-linkage) is suitable for his/her dataset[1]. A hierarchical clustering algorithm outputs a clustering tree. Every pruning of the tree is a clustering of the original dataset. He/she would now like to know which amongst these clustering algorithms is suitable for his task. After having fixed the algorithm, the practitioner would then like to know which amongst these many prunings he/she should chose.

The framework of restricted correlation clustering is applicable in such scenarios. When $\mathcal{F} = \{T\}$ where T is a hierarchical clustering of X, the goal of RCC is to find the pruning from the tree T which has minimum normalized correlation loss. When $\mathcal{F} = \{T_1, \ldots, T_s\}$ where each T_i is a hierarchical clustering of X. Then the goal of RCC is to find a pruning with minimum loss amongst the prunings of all the s trees. Note that finding the pruning of the tree is the same as choosing the stopping point criteria when

[1] A nice overview of hierarchical clustering techniques can be found in [**?**]

running linkage-based algorithms. Hence, the framework can help us choose the right stopping point for a particular hierarchical clustering algorithm.

If $\mathcal{F} = \{C_1, \ldots, C_s\}$ where each C_i is a clustering of the set X then the goal is to find a clustering with minimum loss. Note that $\mathcal{F}$ can be any of the examples as defined above or a union of these or some other finite class.

1.2 Solution strategy

In the RCC framework, we wish to minimize the loss which depends on the unknown target clustering C^*. However, in the absence of any information about C^*, there is no hope to find a clustering that minimizes L_{C^*}. Hence, to solve the RCC problem we allow the clustering (or learning) algorithm to make queries to a C^*-oracle.

Our goal is to calculate quantities $L_{P^+}(C)$ and $L_{P^-}(C)$ (Defn. **??**) for each of the clusterings $C \in \mathcal{F}$ and then choose the clustering with minimum loss. To calculate both these quantities exactly, for each pair of points in our dataset, we would need to know whether they belong to the same-cluster or different-cluster. In other words, we would need access to the complete ground truth clustering C^*. Thus, instead of calculating these two quantities exactly we want to estimate them from a small sample, sampled according to the distributions P^+ and P^-.

One strategy to estimate $L_{P^+}(C)$ (and L_{P^-}) could be the following. Sample a set S_+ (and S_-) of pairs using the distribution P^+ (and P^-). Compute the fraction of mistakes made by each clustering C on S_+ (and S_-). Using the standard results from vc-dimension theory (Thm. **??**), it is known that using this procedure we can estimate L_{P^+} for each of the clusterings $C \in \mathcal{F}$. Similarly, we could also estimate L_{P^-}. Using the two estimates, we could then estimate the loss L_{C^*} for each of the clusterings in our class and choose the clustering which has the smallest loss.

The main problem in this approach is that the distributions P^+ and P^- are unknown (as the target clustering C^* is not known). In Section 2, we discuss two approaches which (approximately) sample according to these distributions. Then in Section 3, we show how these sampling procedures can be used to estimate L_{C^*} for all the clusterings in our class $\mathcal{F}$.

2 Sampling for RCC

We first describe the procedure $\mathcal{P}_0$ which samples according to P^-. Then we describe the procedure $\mathcal{P}_1$ which samples approximately according to the distribution P^+.

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

Input: A set X and a C^*-oracle.
Output: (x, y) 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
```

The procedure samples a pair uniformly at random. Then using the oracle it checks if the sampled pair is negative and terminates if such a pair is found. If not then the process is repeated again.

Lemma 2. *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 $= p + (1-q)p + (1-q)^2p + \ldots = p\sum_{i=0}^{\infty}(1-q)^i = \frac{p}{q} = \frac{U^2(x,y)}{U^2(X^{[2]-})} = P^-(x, y)$. □

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 3. *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 3 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.

2.1 Sampling positive pairs for general metrics

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^+.

The sampling algorithm is described in Alg. **??**. 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.

Algorithm 2: Sampling procedure $\mathcal{P}_{11}$ for positive pairs (general metrics)

Input: A set X, a C^*-oracle and a parameter λ.
Output: One pair $(x, y) \in X^{[2]}$ such that $\mathcal{C}^*(x, y) = 1$

Pre-compute: For all $x \in X$, compute $S_x := \{y : d(x, y) \leq \lambda\}$.
while *TRUE* **do**
    Sample $x \in X$ with probability $\propto |S_x|$.
    Sample y uniformly at random from S_x.
    if $\mathcal{C}^*(x, y) = 1$ **then**
        Output (x, y).
    end
end

Theorem 4. *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 $\mathcal{P}_{11}$ induces a*

distribution T over $X^{[2]}$ such that for any labelling function h over $X^{[2]}$ we have that

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

Note that to sample one positive pair, procedure $\mathcal{P}_{11}$ might need to ask more than one same-cluster query. However, since the metric d is β-informative, we 'expect' the number of 'extra' queries to be 'small'.

Lemma 5. *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 $\mathcal{P}_{11}$ 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. $\mathbf{E}[q] = \frac{1}{p} \leq \frac{1}{\beta}$. □

2.2 Sampling positive pairs for LSHable metrics

The strategy in the previous section was to construct a set $K = \{(x,y) : d(x,y) \leq \lambda\}$ and then sample uniformly from the set K till a positive sample is found. Since most of the positive pairs have distance $\leq \lambda$, this sampling procedure approximates P^+ (the uniform distribution over the set of true positives). However, constructing the set K requires $\Theta(|X|^2)$ time. In this section, we show that if the metric d has some additional structure (is hashable) then we can reduce the pre-processing time to $O(|X|)$. We develop a sampling procedure $\mathcal{P}_{12}$ using techniques from locality sensitive hashing (LSH) combined with rejection sampling. We will show that $\mathcal{P}_{12}$ needs only linear pre-processing time (to build the hash maps) and outputs a positive pair sampled approximately according to P^+.

Locality Sensitive Hashing (LSH)

Before we describe our technique, we introduce some relevant notation. A hash function $h : X \to \mathbf{N}$ maps the set X onto the set of natural numbers. Thus, a hashing function partitions the input of size n into $m \leq n$ different buckets (or blocks) $B_1, \ldots, B_m$ where each $B_i = \{x : h(x) = b_i\}$ for some b_i. Given (X,d), a Locality Sensitive Hashing (LSH) scheme w.r.t the distance metric d (or a similarity metric) aims to partition X into buckets such that

'similar' items map to the same bucket with high probability and 'dissimilar' items end up in different buckets with high probability. For example, MinHash scheme w.r.t Jaccard similarity measure [**?**, **?**] is a common LSH-based hashing scheme. Another example is SimHash scheme w.r.t hamming similarity measure [**?**].

Definition 6 (LSH-based hashing algorithm)**.** *Given a set (X, d) and parameter s. An LSH-based hashing algorithm (or scheme) $\mathcal{A}$ outputs s different partitions $P_1, \ldots, P_s$ of X. Denote $P_i = \{B_{i1}, \ldots, B_{in_i}\}$. We say that $\mathcal{A}$ is (ϵ, ϵ')-tight w.r.t d and λ, λ' if*

- *If $d(x, y) \leq \lambda$ then $\mathbf{P}[b(x, y) = 1] > 1 - \epsilon$*
- *If $d(x, y) > \lambda'$ then $\mathbf{P}[b(x, y) = 1] < \epsilon'$*

where $b(x, y) = 1$ if and only if x, y are together in atleast one of the blocks B_{ij}.

Infact, we show that by choosing s (and other parameters) appropriately, we can construct LSH schemes which are $(\epsilon, \epsilon' = s\ln(1+\epsilon))$-tight w.r.t λ and $\lambda' = 2\lambda\ln(1+1/\epsilon)$. Thus, for simplicity of notation, we say that $\mathcal{A}$ is ϵ-tight w.r.t λ to mean that it is (ϵ, ϵ')-tight w.r.t λ, λ' as chosen above.

Throughout the remainder of this section, we will assume that the hashing scheme satisfies ϵ-tightness. In the appendix, we provide details about why this assumption is justified. However, these results are orthogonal to the current discussion. Hence, we omit it here and only include it in the appendix (Thm. **??**).

We now describe our sampling procedure. Let $\mathcal{B} := \{P_1, \ldots, P_s\} = \{B_{ij} : 1 \leq i \leq s, 1 \leq j \leq |P_i|\}$ be the set of blocks outputted by the hashing scheme and let $Q := \{(x, y) \in B_{ij}\}$. We first choose a block $B \in \mathcal{B}$ with probability proportional to $|B|^2$ (the number of pairs in the block). Then we sample a pair uniformly at random from this block B. Note that this strategy doesn't give us a uniform sample from Q. This is because a pair (x, y) may be present in multiple blocks. To get the uniform sample, we reject the pair with probability inversely proportional to $a(x, y)$ (the number of blocks in which x, y are together). This approach based on rejection sampling ensures that we have a uniform sample from Q.

Next, we check if the pair satisfies $d(x, y) \leq \lambda$. Note that the LSH-based scheme tries to put similar points in the same bucket, hence the probability of

Algorithm 3: Sampling procedure $\mathcal{P}_{12}$ for positive pairs

Input: A set $\mathcal{X}$, a hashing algorithm $\mathcal{A}$, a C^*-oracle and parameter λ.

Output: (x, y) such that $C^*(x, y) = 1$

Pre-compute:

Use an LSH-based hashing scheme $\mathcal{A}$ to obtain partitions $\{P_1, \ldots, P_s\}$.

$\mathcal{B} := \{P_1, \ldots, P_s\} = \{B_{ij} : 1 \leq i \leq s, 1 \leq j \leq |P_i|\}$.

Sampling:

while *TRUE* **do**
  Sample a block B from $\mathcal{B}$ with probability $\propto |B|^2$.
  Sample (x, y) uniformly at random from B^2.
  Let $a(x, y) = \{(x, y) \in B^2 : B \in \mathcal{B}\}$.
  Sample u uniformly at random from $[0, 1]$.
  if $u > \frac{1}{|a(x,y)|}$ **then**
    continue.
  end
  if $d(x, y) \leq \lambda$ *and* $C^*(x, y) = 1$ **then**
    Output (x, y).
  end
end

success at this step is 'high'. Finally, we check if $C^*(x, y) = 1$. Our sampling procedure $\mathcal{P}_1$ is described in Alg. 3.

Thm. 7 shows that with high probability the procedure $\mathcal{P}_{12}$ samples a pair according to a distribution $\mathcal{T}$ which approximates P^+.

Theorem 7. *Given (X, d), a C^*-oracle and parameter λ. Let d satisfy (α, β)-informative w.r.t C^*. Let the hashing algorithm $\mathcal{A}$ satisfy ϵ-tightness w.r.t λ. Then with probability atleast $1-\exp(-2(\nu(1-\alpha)|X_+^2|)^2)$ (over the randomness in the hashing algorithm), $\mathcal{P}_{12}$ samples pairs (x, y) according to distribution $\mathcal{T}$ over $X^{[2]}$ such that for any labelling function $C : X^{[2]} \to \{0, 1\}$, we have*

that

$$\begin{aligned}&\mathop{\mathbf{P}}_{(x,y)\sim P^+}\big[C(x,y)=0\big]-\alpha-\epsilon-\nu\leq\mathop{\mathbf{P}}_{(x,y)\sim T}\big[C(x,y)=0\big]\\&\leq(1+2\nu)(1+2\alpha)\mathop{\mathbf{P}}_{(x,y)\sim P^+}\big[C(x,y)=0\big]\end{aligned}$$

To sample one same-cluster pair, we might need to make more than one same-cluster query to the C^*-oracle. Lemma 8 shows that with high probability, the number of queries made by $\mathcal{P}_{12}$ to sample one positive pair is upper bounded by a small constant.

Lemma 8. *Given set X, a C^*-oracle and parameter λ. Let d be (α,β)-informative w.r.t λ and C^*. Let $\mathcal{A}$ satisfy ϵ-tightness w.r.t λ. Let q be the number of same-cluster queries made by $\mathcal{P}_{12}$. Then with probability atleast $1-\exp(-\nu^2(1-\alpha)^2|X_+^2|^2)$ (over the randomness in the hashing algorithm)*

$$\mathbf{E}[q]\leq\frac{1}{\beta(1-\epsilon-\nu)}$$

The pre-compute stage uses a hashing algorithm to obtain s different partitions. From the discussion in the appendix (Thm. **??**), we its easy to see that this runs in $O(n)$ time. Next, we analyse the time taken to sample one same-cluster pair. Thm. 9 shows that under reasonable assumptions, the time taken is upper bounded by a constant with high probability.

Theorem 9. *Given set X, a C^*-oracle and parameter λ. Let d be (α,β)-informative w.r.t λ and C^*. Let $\mathcal{A}$ satisfy ϵ-tightness w.r.t λ.*

Define $\lambda'=2\lambda\log(1+\frac{1}{\epsilon})$ and $\epsilon'=\lceil\log(\frac{1}{\epsilon})\rceil(1+\log(\frac{1}{\epsilon}))$. Let $K=\{(x,y):d(x,y)\leq\lambda\}$ is the set of all pairs of points with distance $\leq\lambda$. Similarly, define sets $K_1=\{(x,y):\lambda<d(x,y)\leq\lambda'\}$ and $K_2=\{(x,y):d(x,y)>\lambda'\}$. Let $|K_1|\leq\rho_1|K|$ and $\epsilon'|K_2|\leq\rho_2|K|$.

Let t be the time taken to sample one point by procedure $\mathcal{P}_{12}$. Then with probability atleast $1-\exp\left(\frac{-\nu^2(1-\epsilon)(1-\alpha)|X_+^2|}{2}\right)-\exp\left(\frac{-\nu^2\rho_2|K|}{3}\right)$ (over the randomness in the hashing algorithm), we have that

$$\mathbf{E}[t]\leq s^2(1+\frac{1}{\eta})$$

where $\eta:=\frac{(1-\nu)(1-\epsilon)\beta}{(1+\nu)(1+\rho_1+\rho_2)}$.

3 Sample and query complexity of RCC

In the previous section we saw how to sample (approximately) according to the distributions P^+ and P^-. We sample a 'small' set of true positive (or same-cluster) and true negative (or different-cluster) pairs using our distributions. We then choose the clustering $\hat{C} \in \mathcal{F}$ with the minimum number of mistakes on the sampled pairs. We prove that the true normalized correlation loss $L_{C^*}(C)$ is close to the loss of $\hat{C}^*$ (the clustering with minimum loss in $\mathcal{F}$). Thus, our solution strategy shows that by only having a small amount of information about C^* (making a small number of queries) we can find a clustering which is close (in terms of loss) to the optimal clustering in $\mathcal{F}$. We describe this procedure in Alg. 4.

Note that in this section, we have assumed that procedure $\mathcal{P}_{11}$ is used for sampling positive pairs. Similar results can be obtained when instead procedure $\mathcal{P}_{12}$ is used for sampling positive pairs. However, we include those results only in the appendix.

Algorithm 4: Empirical Risk Minimization

Input: (X, d), a set of clusterings $\mathcal{F}$, a C^*-oracle, parameter λ and sizes m_+ and m_-.

Output: $C \in \mathcal{F}$

1 Sample a sets S_+ and S_- of sizes m_+ and m_- using procedures $\mathcal{P}_{11}$ and $\mathcal{P}_0$ respectively.

2 For every $C \in \mathcal{F}$, compute

$$\hat{P}(C) = \frac{|\{(x,y) \in S_+ : C(x,y) = 0\}|}{|S_+|}$$
$$\hat{N}(C) = \frac{|\{(x,y) \in S_- : C(x,y) = 0\}|}{|S_-|}$$

3 Define $\hat{L}(C) = \mu\hat{P}(C) + (1-\mu)\hat{N}(C)$.

4 Output $\arg\min_{C \in \mathcal{F}} \hat{L}(C)$

Thm. 10 analyses the sample complexity of our approach. That is, the number of labelled positive and negative pairs our algorithm needs as input, so that the estimates of the loss based on this sample are close to their true values. We show that as long as the number of sampled pairs are in

$O(\frac{\text{VC-Dim}(\mathcal{F})}{\epsilon^2})$ then our algorithm finds a clustering $\hat{C}$ which is close to the best clustering in $\mathcal{F}$. Here, VC-Dim is a combinatorial property which measures how 'complex' or rich the class of clusterings is. Note that the number of samples needed is independent of the size of the dataset X.

For common classes, like $\mathcal{F} = \{T_1, \ldots, T_s\}$ where each T_i is a hierarchical clustering of X, [**?**] showed that the VC-Dim($\mathcal{F}$) is in $o(\log s)$. Thus for such classes a small number of samples suffice to find a clustering which is close to the best clustering in $\mathcal{F}$.

Theorem 10. *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. 4 and $\hat{C}$ be the output of $\mathcal{A}$. If*

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

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

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

Proof. Let T_0 be the distribution induced by $\mathcal{P}_0$ and T_1 be the distribution induced by $\mathcal{P}_{11}$. Denote by $E(h) = \mathop{\mathbf{P}}_{(x,y)\sim P^+}[h(x, y) = 0]$ and by $G(h) = \mathop{\mathbf{P}}_{(x,y)\sim P^-}[h(x, y) = 1]$.

Using Thm. **??**, we know that if $m_+ > a\frac{\text{VC-Dim}(\mathcal{F})+\log(\frac{1}{\delta})}{\epsilon^2}$ then with probability atleast $1 - \delta$, we have that for all h

$$\begin{aligned} &|\hat{E}(h) - \mathop{\mathbf{P}}_{(x,y)\sim T_1}[h(x, y) = 0]| \leq \epsilon \\ \implies &\hat{E}(h) \leq \epsilon + \mathop{\mathbf{P}}_{(x,y)\sim T_1}[h(x, y) = 0] \leq \epsilon + 2\alpha + E(h) \quad \text{and} \\ &E(h) - 2\alpha - \epsilon \leq \hat{E}(h) \end{aligned} \tag{3}$$

Note that we obtain upper and lower bounds for $\mathop{\mathbf{P}}_{(x,y)\sim T_1}[h(x, y) = 0]$ using Thm. 4. Similarly, if $m_- > a\frac{\text{VC-Dim}(\mathcal{F})+\log(\frac{1}{\delta})}{\epsilon^2}$, then with probability atleast $1 - \delta$, we have that for all h,

$$\begin{aligned} &|\hat{G}(h) - \mathop{\mathbf{P}}_{(x,y)\sim T_0}[h(x, y) = 1]| \leq \epsilon \\ \implies &\hat{G}(h) \leq \epsilon + G(h) \quad \text{and} \quad G(h) - \epsilon \leq \hat{G}(h) \end{aligned} \tag{4}$$

Combining Eqns. 3 and 4, we get that with probability atleast $1-2\delta$, we have that for all $C \in \mathcal{F}$

$$\begin{aligned}
&\hat{L}(C) \leq \mu[\epsilon + E(h) + 2\alpha] + (1-\mu)(\epsilon + G(h)) \\
&\leq L(h) + \epsilon + 2\alpha \\
&\text{And } \hat{L}(C) \geq \mu(E(h) - \epsilon - \alpha) + (1-\mu)(G(h) - \epsilon) \\
&\geq L(h) - \epsilon - \alpha
\end{aligned}$$

Now, let $\hat{C}$ be the output of $\mathcal{A}$ and let $\hat{C}^*$ be $\arg\min_{C \in \mathcal{F}} L(C)$. Then, we have that with probability atleast $1-2\delta$

$$L(\hat{C}) \leq \hat{L}(\hat{C}) + \alpha + \epsilon \leq \hat{L}(\hat{C}^*) + \alpha + \epsilon \leq L(\hat{C}^*) + 2\epsilon + 3\alpha$$

Choosing $\epsilon = \frac{\epsilon}{2}$ and $\delta = \frac{\delta}{2}$ throughout gives the result of the theorem. □

Finally, we analyse the query complexity of our approach. That is the number of queries that our algorithm makes to the C^*-oracle. Our algorithm makes queries during the sampling procedure. We see that to sample m_- negative and m_+ positive pairs the number of queries is 'close' to $m_+ + m_-$ with very high probability. Thus, the number of 'wasted' queries is small.

Theorem 11. *[Query Complexity] Let the framework be as in Thm. 10. With probability atleast* $1 - \exp\left(-\frac{\nu^2 m_-}{4}\right) - \exp\left(-\frac{\nu^2 m_+}{4}\right)$ *over the randomness in the sampling procedure, the number of same-cluster queries* q *made by* $\mathcal{A}$ *is*

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

Proof. Let q_+ denote the number queries to sample the set S_+. We know that $\mathbf{E}[q_+] \leq \frac{1}{\beta}$. Given that the expectation is bounded as above, using Thm. **??**, we get that $q_+ \leq \frac{(1+\nu)m_+}{\beta}$ with probability atleast $1 - \exp(\frac{-\nu^2 m_+}{4})$. Similarly, we get that with probability atleast $1 - \exp(\frac{-\nu^2 m_-}{4})$, $q_- \leq \frac{(1+\nu)m_-}{(1-\gamma)}$. □

3.1 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 12. *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 [?].

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 13 (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 14. *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 $\frac{\sqrt{n}!}{\lfloor\sqrt{n}/2\rfloor!\ \ 2^{\lfloor\sqrt{n}/2\rfloor}} \geq s$

4 Computation complexity of RCC for common clustering classes

Alg. 4 described the general ERM approach to find the 'best' clustering for any class $\mathcal{F}$ of clusterings. Consider the case when $\mathcal{F} = \{C_1, \ldots, C_s\}$ (a finite list of clusterings). Then to implement Alg. 4, we first sample a set $S \subseteq X^2$ and then compute the error of each clustering on this set S. Computing the error of a clustering on S takes $\Theta(|S|)$ time. Hence, for finite $\mathcal{F}$, the ERM approach can be implemented in time $\Theta(|\mathcal{F}||S|)$.

Now, let's focus on the case when $\mathcal{F} = \{T_1, \ldots, T_s\}$ (a finite set of clustering trees). Each tree T_i can contain exponentially many clusterings. Hence, it is not clear if we can still implement the ERM approach in polynomial-time. Consider the problem of implementing the ERM approach when $\mathcal{F} = \{T\}$.

Algorithm 5: ERM approach for a hierarchical clustering tree

Input: A set X, a set $S \subseteq X^2$ labelled according to C^*. Given a clustering tree T on $S_u = \{x : (x, y) \text{ or } (y, x) \in S\}$.

Output: A clustering $\hat{C} \in T$ which implements ERM over T.

Initialize $e(\nu) = a(\nu) = 0$ for all the leaf nodes ν.
for *all non-leaf nodes ν (in a bottom-up manner)* **do**
    Let ν_l be left sub-tree and ν_r the right sub-tree.
    Initialize $s_a = d_a = 0$.
    for $(x, y) \in S$ **do**
        if $x, y \in nl(\nu)$ *and* $C^*(x, y) = 0$ **then**
            $d_a = d_a + 1$.
        end
        if $!(x \in nl(\nu_l)$ *and* $y \in nl(\nu_r))$ *and* $!(y \in nl(\nu_l)$ *and* $x \in nl(\nu_r))$ **then**
            continue
        end
        if $C^*(x, y) = 1$ **then**
            $s_a = s_a + 1$.
        end
    end
    $a(\nu) = d_a$
    $e(\nu) = \min\{e(\nu_l) + e(\nu_r) + s_a, a(\nu)\}$
end

The goal is to find the pruning in the tree which minimizes the loss $\hat{L}$. Given tree T, the clustering with the smallest loss is either the clustering which assigns all the points to a single cluster or the clustering with the best loss in T_l (left subtree) concatenated with the clustering with the best loss in T_r (right subtree). Before we describe our approach lets introduce a bit of notation.

For every node ν in the hierarchical clustering tree, let $nl(\nu)$ be the leaves which are descendants of ν. For a leaf node ν, $nl(\nu) = \{\nu\}$. Let $\mu_1 = \mu|S_-|$ and $\mu_2 = (1 - \mu)|S_+|$. Let

$$e(\nu) = \arg\min_{C \in T_\nu} \mu_1 |S_+| \hat{P}(C) + \mu_2 |S_-| \hat{N}(C)$$

where T_ν is the tree T restricted to the descendants of ν. Let $a(\nu) = \mu_1|S_+|\hat{P}(C) + \mu_2|S_-|\hat{N}(C)$ where C is the clustering which assigns all the descendants of ν to a single cluster. We are now ready tp describe our bottom-up approach in Alg. 5. Its easy to see that the running time of the approach is $\Theta(|S|^2)$.