Intractability and Clustering with Constraints

Ian Davidson DAVIDSON@CS.ALBANY.EDU
S.S. Ravi RAVI@CS.ALBANY.EDU
Department of Computer Science, State University of New York, 1400 Washington Ave, Albany, NY 12222

Abstract

Clustering with constraints is a developing area of machine learning. Various papers have used constraints to enforce particular clusterings, seed clustering algorithms and even learn distance functions which are then used for clustering. We present intractability results for some constraint combinations and illustrate both formally and experimentally the implications of these results for using constraints with clustering.

1. Introduction, Motivation and Previous Results

Clustering is used extensively in unsupervised machine learning applications such as information retrieval and natural language understanding [Wagstaff et. al. 2001]. Recent work published in the machine learning and data mining community has investigated the introduction of supervision into clustering in the form of instance-level constraints which effectively provide hints to the composition of a desirable clustering of the instances. This line of work has followed a natural progression since Wagstaff and Cardie's seminal paper [Wagstaff and Cardie 2000] which we now discuss briefly.

Clustering Under Constraints. In 2000 Wagstaff and Cardie [Wagstaff and Cardie 2000] introduced to the machine learning community the most common form of constraints: must-link (ML) where two instance must be in the same cluster and cannot-link (CL) where they must be in different clusters. Their COP-k-means algorithm attempts to find a set partition that minimizes the vector quantization error and also satisfies *all* of the constraints. The ML and CL constraints in conjunction offer the ability to incorporate strong background knowledge into the clustering process with respect to the type of clusters to be found. For example when clustering automobile GPS trace information to form clusters (traffic lanes) [Wagstaff et. al. 2001], the physical distance between lanes (4 meters) can be used to generate CL constraints between instances. In academic situations constraints are typically randomly generated from small amounts of labeled data. If two randomly chosen instances have the same (different) label an ML (a CL) constraint is generated between them. However, the constraints may be generated from background knowledge such as in the traffic lane application above. Wagstaff's empirical results in her thesis [Wagstaff 2002] and papers convincingly show that using constraints can improve performance with respect to predicting an extrinsic label over unsupervised clustering.

Seeding Clustering Algorithms With Constraints In 2002 Basu and collaborators [Basu et. al. 2002] explored using constraints to initialize the k-means algorithm. Their core idea is to make use of scarce labeled points to compute approximations to the cluster centroids. This idea can be extended to computing the transitive closure over the must-linked points to generate a series of connected components and then seeding the cluster centroids by the average of the points in each connected component. Their experimental work showed clearly that seeding k-means with the labeled data provides improved performance over just random seeding.

Learning a Distance Function From Constraints and Then Clustering In 2003, Xing and collaborators [Xing et. al. 2003] introduced another use of ML and CL constraints, namely to learn a distance function. The aim of their work is to learn a mapping from the original m dimensional space to another m dimensional space so that the must-linked points are close and the cannot-linked points are far apart. Xing's experimental results show that learning a distance function from the constraints and then clustering the data with the newly learnt distance function but *not enforcing* the constraints typically performs better than just finding a clustering that enforces the constraints.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

Active Constraint Generation In 2004 Basu and collaborators [Basu et. al. 2004a] develop an approach to actively select the most informative constraints. The problem they address is: given an Oracle that has access to a set of labels that implicitly defines a set partition, which subset of points should be selected to reveal their respective constraints on each other. Their experimental results show that actively selecting constraints produces better results than randomly generating constraints by choosing two instances and comparing their labels.

Clustering under Selective Constraints Finally, the PKM [Basu et. al. 2004b] and CVQE [Davidson and Ravi 2005] algorithms allow the ignoring of constraints if their satisfaction leads to a significant worsening of the objective function. We can think of this work as being penalized COP-k-means. When constraints are not helpful (say due to noise), this work shows that ignoring some constraints produces better results than attempting to satisfy all constraints as COP-k-means does.

1.1. Negative Intractability Results

However, when using constraints with clustering, it is quite possible to specify constraints inappropriately particularly when constraints are not generated from labeled data. As a trivial example, no clustering is possible for any value of k under the constraints $ML(a,b)$, $CL(a,b)$. Davidson and Ravi [Davidson and Ravi 2005] discussed that even for self-consistent constraints set such as $CL(a,b)$, $CL(b,c)$ and $CL(a,c)$, there is no clustering for $k \leq 2$. Furthermore, their worst-case complexity results show that to solve the clustering problem under cannot and must link constraints involves solving a sub-problem that is **NP**-complete (intractable). They discuss how algorithms that cluster to satisfy all constraints such as COP-k-means (i.e. *Cluster Under Constraints*) at each iteration must indirectly answer the feasibility question: Does there exist any partition of the set of points that satisfies all the constraints? This problem was formally defined as the *feasibility* problem:

Definition 1.1 ***The Feasibility Problem.*** *Given a set D of data points, a collection C of ML and CL constraints on some points in D, upper (K_u) and lower bounds (K_l) on the number of clusters, does there exist at least one partition of D into k clusters such that $K_l \leq k \leq K_u$ and all the constraints in C are satisfied?*

The feasibility problem for clustering under ML constraints is in **P** while clustering under CL only and ML and CL is **NP**-complete, as can be shown by a reduction from graph coloring [Davidson and Ravi 2005]. Thus, Davidson and Ravi conclude that, unless $\mathbf{P} = \mathbf{NP}$, there cannot be an efficient clustering algorithm that satisfies all constraints for all data sets. It is important to note that they are not stating that there does not exist such an algorithm, just that efficient algorithms cannot exist under the common assumption that $\mathbf{P} \neq \mathbf{NP}$. In this way, rather than being an impossibility result such as Kleinberg's impossibility theorem for clustering [Kleinberg 2002], their work provides a computational intractability result.

Contributions of This Paper. This paper makes two main contributions. Firstly, we extend the original intractability results for constrained clustering with related results from graph theory. We then discuss and where appropriate experimentally verify that these intractability results have implications beyond just clustering to satisfy a set of constraints. We show implications for a) Seeding and active constraint generation (section 4) b) Clustering under most constraints (and ignoring some) (section 5), and c) Learning distance functions (section 6).

Importantly, these intractability results and their implications are related. For example, if no clustering algorithm can efficiently satisfy all constraints, then it would seem natural to try to minimally prune the set of constraints so that they can be satisfied efficiently. In this paper we show this problem is also intractable. Similarly, given that we cannot write an efficient algorithm to satisfy all constraints or optimally prune a constraint set, a natural direction is to make sure that k is guessed to correctly match say the number of extrinsic labels used to generate the constraints so that infeasibility cannot occur. We not only show that BIC and AIC do not return the correct number of extrinsic class labels, but also that efficiently computing the value of k from the answers provided by an Oracle is computationally intractable. The paper begins by surveying previous intractability results in section 2 and then introducing our new intractability results in section 3. Sections 4, 5 and 6 build upon the previous results and empirically demonstrate the impact of these intractability results on seeding algorithms using constraints, clustering while ignoring the minimal number of constraints and learning a distance function from constraints after which we conclude and summarize our work.

2. Coloring, Clustering and Constraints

Consider clustering under the set of constraints: $ML(a,b), ML(a,c), ML(d,e), ML(f,g), ML(h,i)$, $ML(j,k), CL(a,e), CL(i,j), CL(d,k), CL(e,l)$. The feasibility problem involves only those instances $a, b, \ldots, l$. We can represent the feasibility problem as a graph as follows.

1. Find connected components $C_1, ..., C_r$ by calculating the transitive closure over the ML constraints.

2. Replace each connected component with a single node and have a single node for each instance that appears only as part of a CL constraint.

3. For each constraint $\mathrm{CL}(x, y)$, place an edge between nodes representing instances x and y.

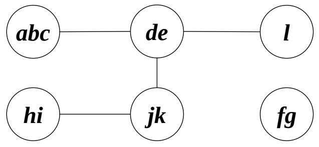
Figure 1 shows the resulting graph for our simple example. To find a feasible solution to satisfy the must-link constraints involves computing the transitive closure in step 1) above and checking whether the number of connected components is at least K_ℓ. Since the transitive closure computation takes time $O(n+m)$, where n is the number of nodes and m the number of edges, the feasibility problem for ML constraints is in **P**. Clustering to satisfy the CL constraints is then a case of assigning each node an integer value from 1 to k so that no two adjacent nodes have the same value. This is at least as hard as the graph k-coloring problem which is known to be **NP**-complete. To find a feasible clustering in our example we can assign instances f and g to any cluster so long as they are together. However, care must be taken to assign the instances *de* so that the assigned value does not conflict with the assignments for l and jk and *abc*.

Importantly, since the feasibility problem is intractable for just CL constraints or ML and CL constraints together, then unless $\mathbf{P} = \mathbf{NP}$, there cannot exist an efficient algorithm that satisfies all the constraints. *Furthermore, if the constraints are generated from k^* different labels and we attempt to cluster the data for k clusters where $k < k^*$, then no feasible solution need exist.* But since the feasibility problem is intractable, we will not be able to tell efficiently that no feasible clustering exists! Most of the previous experimental results reported assume that k^* is known which need not be the case. This is a large assumption and as we shall prove, if k^* is unknown then we cannot efficiently estimate it even if we have access to an Oracle.

3. New Intractability Results

Previous work [Davidson and Ravi 2005] states that in the worst-case, unless $\mathbf{P} = \mathbf{NP}$, no efficient (i.e. polynomial time) clustering algorithm that satisfies all constraints can be developed. We now present related results that we shall later show have an impact on

Figure 1. A graphical representation of *ML(a,b), ML(a,c), ML(d,e), ML(f,g), ML(h,i), ML(j,k), CL(a,e), CL(i,j), CL(d,k), CL(e,l)*



constrained clustering beyond just trying to satisfy all constraints. Our theorems can be summarized as follows.

- Given a collection of ML and CL constraints, the problem of determining the number of extrinsic labels k^* used to generate the constraints is computationally intractable. Furthermore, even efficiently approximating the value of k^* is computationally intractable. We call this the **No Efficient Approximation Theorem** (Theorem 3.1).

- Given a clustering into k clusters that violates some of the constraints, determining whether the clustering can be fixed so that all constraints are satisfied is also computationally intractable. We refer to this as the **No Efficient Repair Theorem** (Theorem 3.2).

- Given a set of constraints for which no feasible solution exists for a particular value of k, identifying the minimal subset of constraints to remove/prune so there is a feasible clustering for the remaining constraints is computationally intractable. We refer to this as the **No Efficient Minimum Pruning Theorem** (Theorem 3.3).

The remainder of this section proves these results and can be skipped on first reading of this paper.

3.1. Some Definitions

3.1.1. Graph Theoretic Definitions

The problems addressed in subsequent sections are closely related to the **minimum coloring** problem for undirected graphs. For the convenience of the reader, formal definitions of coloring and related problems are given below.

Given an undirected graph $G(V, E)$ and an integer $K \le |V|$, the K**-coloring** problem asks whether each node of the graph can be assigned one of at most K colors so that whenever two nodes v_i and v_j of the graph are adjacent (i.e., $\{v_i, v_j\} \in E$), the colors assigned to v_i and v_j are different. Note that K-coloring is a *decision* problem; the answer to the problem is "Yes" or

"No". It is well known that the K-coloring problem is **NP**-complete [Garey and Johnson 1979]. It is widely believed that there are no efficient (i.e., polynomial time) algorithms for any **NP**-complete problem.

In the corresponding optimization problem, called **minimum coloring**, we are given just the graph $G(V, E)$, and we are required to find the smallest integer K such that G is K-colorable. From the computational intractability of K-coloring (i.e., the decision version of the coloring problem), it is readily seen that the minimum coloring problem (i.e., the optimization version) is also computationally intractable. Since the notion of **NP**-completeness applies only to decision problems, the term "**NP**-hard" is used when referring to the intractability of optimization problems.

Suppose a given graph $G(V, E)$ is not K-colorable. One way to achieve K-colorability is to delete some edges[1] of G. This edge deletion problem can be formulated either as a minimization problem or a maximization problem. In both versions, certain edges are deleted from G to ensure that the resulting subgraph is K-colorable. In the minimization version, which we call **minimum edge deletion K-coloring**, the quality of a solution is measured by the number of edges *deleted* from E. In the maximization version, which is known in the literature as the **maximum K-colorable subgraph** problem [Ausiello et. al. 1999], the quality of a solution is measured by the number of edges *retained* in E. Clearly, the **NP**-hardness of the minimum coloring problem implies that of both the maximization and minimization versions of the edge deletion problem.

3.2. Variants of Feasibility: Computational Intractability Results

When the feasibility problem for clustering is **NP**-complete, some options for dealing with the problem were mentioned earlier. In this section, we formally show that many of these options also lead to computationally intractable problems.

A common thread that runs through the results in this section is the following: the minimum graph coloring problem is embedded in the feasibility problem (and its variants) involving CL constraints. The main idea needed to prove the embedding is the following straightforward reduction [Davidson and Ravi 2005] of the minimum coloring problem to the feasibility problem.

[1]As will be seen, deleting edges corresponds to pruning a given constraint set in the case of the feasibility problem for clustering.

Reduction R:

(a) Given a graph $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, construct a collection $S = \{p_1, p_2, \ldots, p_n\}$ of n points. (Point p_i corresponds to node v_i, $1 \leq i \leq n$. Spatial coordinates for the points in S are not specified since they play no role in the reduction.)

(b) For each edge $\{v_i, v_j\} \in E$, add the constraint $\mathrm{CL}(p_i, p_j)$.

Clearly, the feasibility problem produced by the above reduction involves only CL constraints. (The number of CL constraints produced is equal to the number of edges in the graph G.) Reduction R ensures that each color class of G (i.e., subset of nodes with the same color) corresponds to a cluster in a solution to the feasibility problem and vice versa. An easy consequence of the reduction is the following [Davidson and Ravi 2005].

Proposition 3.1 *Let $G(V, E)$ be any undirected graph and let point set S and constraint set C constitute the feasibility problem instance produced by carrying out the reduction R on G. For any integer K, there is a feasible clustering with K clusters if and only if G is K-colorable.* ■

We will use the above proposition several times in the ensuing discussion.

3.2.1. Obtaining a Feasible Clustering with Near-Minimum Number of Clusters

This subsection considers the problem of producing a partition with a minimum number of clusters. Since this problem is **NP**-hard, one may be interested in obtaining a partition that is a ρ-approximation for some $\rho > 1$ (i.e., a partition that uses at most ρ times the smallest number of clusters). However, a computational intractability result[2] for obtaining a near-optimal partition can be easily established as follows.

Theorem 3.1 ***[No Efficient Approximation Theorem]*** *For any $\epsilon > 0$, there is no $O(n^{1-\epsilon})$-approximation for the problem of obtaining a feasible clustering with a minimum number of clusters, unless **NP = ZPP**.*

Proof: If, for some $\epsilon > 0$, there is an $O(n^{1-\epsilon})$-approximation for the feasibility problem, then there

[2]This intractability result is based on the widely believed assumption that complexity classes **NP** and **ZPP** are different. For a definition of complexity classes, we refer the reader to [Papadimitriou 1994].

would be such an approximation for the minimum coloring problem as well. This follows directly from Proposition 3.1. However, it is known [Feige and Kilian 1998] that the existence of such an approximation for the minimum coloring problem would imply that **NP** = **ZPP**. ■

3.2.2. Repairing an Infeasible Clustering

In this subsection, we consider the problem of repairing a given infeasible clustering. In this problem, we are given a point set S, a constraint set C and a partition of S into K clusters S_1, S_2, …, S_K such that one or more of the constraints in C are violated. The goal is to obtain another partition into K clusters S'_1, S'_2, …, S'_K such that none of the constraints in C is violated. We now observe that this problem is also computationally intractable. (The proof essentially points out that the given infeasible clustering is a red herring with respect to the worst-case complexity of the problem.)

Theorem 3.2 ***[No Efficient Repair Theorem]*** *The problem of repairing an infeasible clustering is* ***NP****-complete.*

Proof: The problem of repairing an infeasible clustering is in **NP** since one can guess a partition and verify that all the constraints are satisfied.

To prove **NP**-hardness, consider the following minor variant of the K-coloring problem: Given a graph $G(V, E)$ and an invalid K-coloring, can the coloring be modified into a valid K-coloring? This variant of the K-coloring problem is also **NP**-complete, since regardless of the initial (invalid) coloring, we can get a valid K-coloring if and only if G is K-colorable. By starting with this variant of K-coloring and carrying out Reduction R, one can obtain a feasibility problem along with an infeasible clustering. By Proposition 3.1, the resulting infeasible clustering can be repaired into a feasible solution if and only if the K-coloring problem has a solution. Hence, the problem of fixing an infeasible clustering is also **NP**-complete. ■

3.2.3. Pruning a Constraint Set

When there is no solution to an instance of the feasibility problem, one possible approach to obtain solutions is to *prune* the constraint set, that is, to delete some constraints. Since constraints have the ability to drive clustering algorithms into producing useful partitions, one may be interested in deleting only the smallest number of constraints from the given constraint set to ensure feasibility. This section examines two variants of this pruning problem.

The first variant, which we call the **minimum pruning problem**, can be formulated as follows: Given a set of points S and a constraint set C, delete a minimum cardinality subset C_d of C so that there is a feasible K-clustering for the resulting constraint set $C - C_d$. For this optimization problem, the quality of a pruning algorithm is measured by the number of deleted constraints (i.e., $|C_d|$). One can readily establish an intractability result for this problem as follows.

Theorem 3.3 ***[No Efficient Minimum Pruning]*** *For any $\rho \geq 1$, there is no ρ-approximation algorithm for the problem of deleting the minimum number of constraints, unless* ***P*** *=* ***NP****.*

Proof: First, consider the minimum edge deletion K-coloring problem where we are given a graph $G(V, E)$ and an integer K and the goal is to delete the smallest number of edges in E so that the resulting graph is K-colorable. The smallest number of edges to be deleted is zero if and only if G is K-colorable. Thus, for any $\rho \geq 1$, a ρ-approximation algorithm for this coloring problem cannot delete any edges, if the graph G is K-colorable. Therefore, for any $\rho \geq 1$, the existence of any ρ-approximation algorithm for the minimum edge deletion K-coloring problem would enable us to efficiently decide whether G is K-colorable; that is, it would imply that **P** = **NP**.

By starting with the minimum edge deletion K-coloring problem and carrying out Reduction R, we obtain an instance of the feasibility problem where the goal is to ensure feasibility by deleting the minimum number of constraints. Moreover, for any $q \geq 0$, any solution obtained by deleting q constraints corresponds to deleting q edges to ensure that G is K-colorable. Therefore, for any $\rho \geq 1$, the existence of a ρ-approximation algorithm for the minimum constraint pruning problem would imply a similar approximation algorithm for the minimum edge deletion coloring problem; that is, it would imply that **P** = **NP**. ■

Since the K-coloring problem is **NP**-complete for any *fixed* $K \geq 3$ [Garey and Johnson 1979], it follows from the above proof that the minimum pruning problem has no efficient approximation (modulo the complexity theoretic assumption that **P** ≠ **NP**) even for fixed values of K.

A second variant of the pruning problem involves retaining the *maximum* number of constraints from the given constraint set. This problem, which we call the **maximum feasible subset problem**, can be formulated as follows: Given a set of points S and a constraint set C, obtain a maximum cardinality subset

C_r of C such that there is a feasible K-clustering for the set C_r. For this optimization problem, the quality of a pruning algorithm is measured by the number of constraints that are *retained* (i.e., $|C_r|$). The maximum feasible subset problem is also **NP**-hard since its decision version is equivalent to the decision version of the minimum pruning problem; deleting the minimum number of constraints is equivalent to retaining the maximum number of constraints. In terms of approximability, the two problems behave differently. For example, when the constraint set C contains only CL constraints, there is no efficient approximation algorithm for the minimum pruning problem, unless $\mathbf{P} = \mathbf{NP}$ (Theorem 3.3). Moreover, this result holds even for any fixed $K \geq 3$. The maximum feasible subset problem is equivalent to the maximum K-colorable subgraph problem defined earlier. For this problem, efficient approximation algorithms are known for fixed values of K [Ausiello et. al. 1999].

4. Implications of Intractability Results for Seeding k-Means and Active Constraint Generation

As mentioned earlier, several uses of constraints for clustering have been studied in the literature. Here, some implications of our results for two of these approaches, namely the seeding of clustering algorithms using constraints and active constraint generation, are examined. It should be noted that both of these approaches inherently rely on knowing the value of k^*, the number of extrinsic labels. This is so since if we attempt to cluster with k^* under-estimated then no feasible clustering may exist. Furthermore, as we shall see in later sections, learning distance functions and then clustering with a lower value than k^* produces worse results in terms of cluster accuracy on the extrinsic label than not using any constraints.

It is important to note that we use a pragmatic/relaxed Oracle in this section, meaning that an answer of "no" to the question "Are x and y cannot-linked?" does not mean they are must-linked. This corresponds to the situation where labels are not known for all the points.

Theorem 4.1 ***[Insufficiency of Using Just a Cannot-Link Oracle].*** *Given an Oracle that can only answer 'yes' or 'no' to the question "Is there a CL constraint between instances x and y?", the true value of k^* cannot be determined in polynomial time, unless $\mathbf{P} = \mathbf{NP}$.*

Proof: We will show that if there is such a polynomial time algorithm, it can be used to solve the k-Color problem in polynomial time, contradicting the assumption that $\mathbf{P} \neq \mathbf{NP}$.

# Clusters	AIC	PC	BIC/MDL	ICOMP
1	0 (0)	1 (0)	2 (0)	0 (0)
2	28 (0)	14 (0)	29 (0)	0 (0)
3	32 (0)	46 (0)	28 (0)	7 (0)
4	19 (1)	18 (0)	24 (2)	23 (0)
5	2 (55)	12 (11)	14 (41)	29 (2)
6 (Glass)	12 (12)	7 (32)	3 (28)	32 (7)
7	7 (10)	2 (38)	0 (16)	10 (72)
8	5 (12)	0 (11)	0 (13)	0 (18)
9	3 (7)	0 (8)	0 (0)	0 (1)
10 (Digit)	3 (2)	0 (0)	0 (0)	0 (0)

Table 1. The appropriate number of clusters for the Glass and in parentheses the Digit data sets for a variety of model complexity measures. For each value of k EM was used to estimate the maximum likelihood and parameters. The value of k^* for Glass is 6 and for Digit 10.

Consider an instance of the k-Color problem, given by a graph $G(V, E)$ and an integer k. As before, the set S of the points to be clustered is one-to-one correspondence with the set of nodes. For each query of the form "Is there a CL constraint between points x and y?", the Oracle responds "Yes" if the there is an edge in G between the two nodes corresponding to the points x and y; otherwise, the oracle responds "No". Now, suppose the algorithm that uses the Oracle is able to determine in polynomial time the smallest value k^* for which there is a feasible clustering satisfying all the given CL constraints. Then, by simply checking whether $k \geq k^*$, one can determine whether or not G is k-colorable. This contradicts the assumption that $\mathbf{P} \neq \mathbf{NP}$. ■

The above result shows using just an Oracle for CL constraints, the problem of determining k^* is computationally intractable. As the following observation points out, using an Oracle for just ML constraints is adequate for determining k^*. The idea is that we can consult the Oracle for each pair of points, thus resulting in a total of $\binom{n}{2} = O(n^2)$ queries, and use the results to construct a valid partition with k^* blocks. Unfortunately, when the number of points to be clustered is large, the number of queries used by this simple approach is impractical.

Observation 4.1 ***[Sufficiency of Using Just a Must-Link Oracle].*** *Given an Oracle that can only answer 'Yes' or 'No' to the question "Is there an ML constraint between points x and y?", the true value of k^* (the number of extrinsic labels) can be determined using $O(n^2)$ queries, where n is the number of points to be clustered.* ■

Figure 2. A Graph of the average (500 individual results) number of connected components over randomly created sets of ML constraints of varying size. Similar results exist for Pima, Ion and Iris datasets.

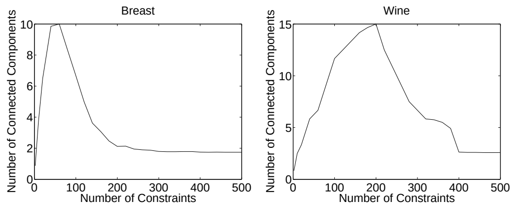
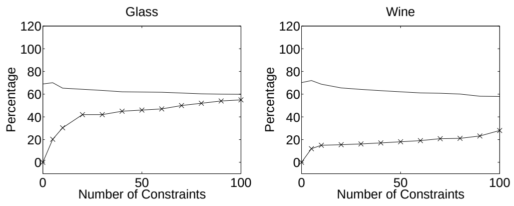


Figure 3. Results over 500 randomly created sets of ML and CL constraint sets of varying size (x-axis). The y-axis shows accuracy (Solid Line) measured according to Rand index and proportion of constraints ignored (Crossed Line) in final clustering.



The results of this theorem can be illustrated as shown in Figure 2. Initially the number of connected components increases rapidly but then as the transitivity and entailment properties of ML and CL constraints [Basu et. al. 2004b] become prevalent the number decreases to the number of extrinsic labels (k^*). It is then tempting to make use of the transitive nature of ML constraints and the entailment property of CL constraints (i.e. $ML(a,b), ML(c,d)$ and $CL(a,c) \rightarrow CL(a,d), CL(b,c)$ and $CL(b,d)$) [Basu et. al. 2004b] to find a value of k^* using a small number of queries. However the No Efficient Approximation Theorem (Theorem 3.1) means that determining k^* from both ML and CL constraints is also computationally intractable. Another alternative is to try to determine k from the unlabeled points. There are a variety of methods of determining the number of clusters such as Aikike information criterion (AIC), Bayesian Information Criterion (BIC) and the closely related Minimum Description Length (MDL) criterion, Partition Coefficient (PC) and ICOMP criterion. See [Oliver et. al. 1996] for explanations and formulas. However, Table 4 shows that for data sets with large numbers of extrinsic labels, all criteria typically provide a value of k that is significantly less than the true value on average. This is to be expected as many model complexity estimators attempt to embody Occam's razor and hence favor simpler models. This result is consistent with prior work [Oliver et. al. 1996] where it is observed that when clusters overlap significantly, AIC, PC, MDL and ICOMP tend to underestimate the number of clusters. However, generating a set of constraints from labels with k^* different values and then attempting to find a clustering with k clusters where $k < k^*$ may fail since a feasible clustering may not exist particularly as the number of constraints increases.

5. Implications for Algorithms That Attempt to Ignore Constraints

Given that satisfying all constraints is computationally intractable, there are two viable alternatives: (a) ignoring some constraints while clustering and then trying to *fix* the clustering so that all constraints are satisfied and (b) initially ignoring the constraints and finding the best clusterings for k and then selecting the *fewest* constraints to prune so that the clustering is feasible. Unfortunately, one cannot expect to carry out either (a) or (b) efficiently due to the No Efficient Repair Theorem (Theorem 3.2) and the No Efficient Pruning Theorem (Theorem 3.3) respectively.

6. Implications for Learning Distance Functions and Then Clustering

The results of Xing and collaborators [Xing et. al. 2003] and Basu and collaborators [Basu et. al. 2004b] showed that learning a distance function from constraints and then using this distance function to cluster the data and *not* enforcing the constraints produced better results than just enforcing the constraints. This is to be expected since by learning a distance function from say $ML(x,y)$ we not only make x and y be close but all points surrounding x and y. However, their experimental results are limited to clustering the data when the value of k is set to be the number of extrinsic labels (k^*) and hence a feasible solution always exists. In Figure 3 we show the results of similar experiments for data sets where $k > 2$ except we cluster the data for k being one less than the number of extrinsic labels. In this way, no feasible solution need exist for some constraint sets generated. Our results show that in this situation the accuracy on the data sets can increase, but overall, constraints can adversely affect the performance of the clustering algorithm. The leftmost point on these plots indicate clustering with no constraints. Furthermore, we see (crossed line) that when evaluating the clustering, most of the constraints that were used to learn the distance function are effectively ignored in that two must-linked (cannot-linked) points are placed in different (the same) clusters. We can formalize this empirical observation in the following theorem.

Theorem 6.1 ***[Futility of Learning a Distance Function and Then Clustering for an Infeasible Value of k Theorem].*** *Let S be a set of data points and let C be a collection of ML and CL constraints over the set S. Let $k^* \geq 2$ denote the minimum number of clusters needed to satisfy all the constraints in C. Suppose an algorithm learns a distance function for S from the constraint set C and the learnt distance function satisfies the following* ***distance criterion****: the maximum distance between any pair of points involved in an ML constraint is* less than *the minimum distance between any pair of points involved in a CL constraint. Then, any clustering of S with* less than *k^* clusters will violate the distance criterion.*

Proof. Since k^* is the minimum number of clusters needed to satisfy all the constraints in C, it can be seen that any partition of S into $k^* - 1$ or fewer clusters violates at least one CL constraint; that is, there is a pair of points x and y such that C contains the constraint $CL(x, y)$, but x and y are in the same cluster. (In other words, the chosen partition with less than k^* clusters enforces an ML constraint on x and y.) Since the learnt distance function satisfies the distance criterion, the distance between x and y is larger than the distance between any pair of points which are involved in an ML constraint. However, x and y are in the same cluster. So, in the clustering with less than k^* clusters, the distance criterion is violated. ■

7. Conclusion

We have built upon earlier intractability results that showed that finding a feasible solution to the problem of clustering under a set of constraints is intractable. We have added several new intractability results. Firstly, given a set of constraints, efficiently calculating even an approximation to the minimum number of extrinsic labels the constraints were generated from is intractable (Theorem 3.1). Secondly, given a set of constraints and a clustering that does not satisfy all of the constraints, the problem of fixing this clustering to satisfy all the constraints is intractable (Theorem 3.2). Thirdly, given a set of constraints and a value of k for which there exists no feasible clustering, the problem of minimally pruning the constraint set so that there is a feasible clustering with k clusters is also intractable. We then showed that these results have implications for several uses of constraints in clustering by presenting empirical results where appropriate. For example, we showed that if a distance function is learnt from a set of constraints and then the points are clustered using a value of k for which no feasible solution need exist, then the vast majority of constraints will effectively be ignored in the final clustering and that the learnt distance function generally performs worse than the Euclidean distance function. We note that these implications were most likely not noted in earlier works which clustered the data using the same number of clusters as extrinsic labels (k^*) used to generate the constraints. In such a situation, feasible solutions will always exist.

Acknowledgments

We gratefully acknowledge NSF funding for this work under CAREER grant IIS-0643668.

References

[Ausiello et. al. 1999] Ausiello G., et. al., *Complexity and Approximation*, Springer, 1999.

[Basu et. al. 2002] Basu S., Banerjee A. & Mooney R., "Semisupervised Learning by Seeding", ICML 2002.

[Basu et. al. 2004a] Basu S., Bilenko M. & Mooney R., "Active Semi-Supervision for Pairwise Constrained Clustering", SIAM Data Mining, 2002.

[Basu et. al. 2004b] Basu S., Bilenko M. & Mooney R., "A Probabilistic Framework for Semi-Supervised Clustering", ACM KDD, 2004.

[Davidson and Ravi 2005] Davidson I. & Ravi S. S., "Clustering with Constraints: Feasibility Issues", SIAM Data Mining, 2005.

[Feige and Kilian 1998] Feige U. & Kilian J., "Zero knowledge and the chromatic number", *J. Computer and System Sciences*, Vol. 57, 1998, pp. 187–199.

[Garey and Johnson 1979] Garey M. & Johnson D., *Computers and Intractability*, 1979.

[Kleinberg 2002] Kleinberg J., "An Impossibility Theorem for Clustering", NIPS 2002, pp. 446–453.

[Oliver et. al. 1996] Oliver J., Baxter R. & Wallace C., "Unsupervised Learning Using MML", ICML 1996.

[Papadimitriou 1994] Papadimitriou C., *Computational Complexity*, Addison-Wesley, 1994.

[Wagstaff and Cardie 2000] Wagstaff K. & Cardie C., "Clustering with Instance-Level Constraints", ICML, 2000.

[Wagstaff et. al. 2001] Wagstaff K., Cardie C, Rogers S. & Schroedl S., "Constrained k-means Clustering with Background Knowledge", ICML, 2001.

[Wagstaff 2002] Wagstaff K., *Intelligent Clustering with Instance-Level Constraints*, Ph.D. Dissertation, Cornell University, 2002.

[Xing et. al. 2003] Xing E., Ng A., Jordan M. & Russell S., "Distance Metric Learning, with Application to Clustering with Side-Information", NIPS, 2003.