

Value, Cost, and Sharing: Open Issues in Constrained Clustering

Kiri L. Wagstaff

Jet Propulsion Laboratory, California Institute of Technology,
Mail Stop 126-347, 4800 Oak Grove Drive, Pasadena CA 91109, USA,
`kiri.wagstaff@jpl.nasa.gov`

Abstract. Clustering is an important tool for data mining, since it can identify major patterns or trends without any supervision (labeled data). Over the past five years, semi-supervised (constrained) clustering methods have become very popular. These methods began with incorporating pairwise constraints and have developed into more general methods that can learn appropriate distance metrics. However, several important open questions have arisen about which constraints are most useful, how they can be actively acquired, and when and how they should be propagated to neighboring points. This position paper describes these open questions and suggests future directions for constrained clustering research.

1 Introduction

Clustering methods are used to analyze data sets that lack any supervisory information such as data labels. They identify major patterns or trends based on a combination of the assumed cluster structure (e.g., Gaussian distribution) and the observed data distribution. Recently, semi-supervised clustering methods have become very popular because they can also take advantage of supervisory information when it is available. The first work in this area proposed a modified version of COBWEB that enforced pairwise constraints indicating when two items were known *a priori* to either belong to the same cluster (must-link) or different clusters (cannot-link) [1]. It was followed by constrained versions of the k-means and EM clustering algorithms [2, 3]. Later work expanded this approach to accommodate soft constraints (preferences) [4, 5] and to infer new distance metrics over a given data set, based on the available constraints. Some metric learning methods are restricted to accommodating must-link constraints only [6], while others can also accommodate cannot-link constraints [7, 8, 5].

These advances have led to further study of the impact of incorporating constraints into clustering algorithms, particularly when applied to large, real-world data sets. Important issues that have arisen include:

1. Given the recent observation that some constraint sets can *adversely* impact performance, how can we determine the utility of a given constraint set, prior to clustering?

2. How can we minimize the effort required of the user, by active soliciting only the most useful constraints?
3. When and how should constraints be propagated or shared with neighboring points?

This paper contributes descriptions of each of these open questions. In identifying these challenges, and the state of the art in addressing them, we highlight several directions for future research.

2 Open Questions

2.1 Value: How Useful is a Given Set of Constraints?

It is to be expected that some constraint sets will be more useful than others, in terms of the benefit they provide to a given clustering algorithm. For example, if the constraints contain information that the clustering algorithm is able to deduce on its own, then they will not provide any improvement in clustering performance. However, virtually all work to date values constraint sets only in terms of the number of constraints they contain. The ability to more accurately quantify the utility of a given constraint set, prior to clustering, will permit practitioners to decide whether to use a given constraint set, or to choose the best constraint set to use, when several are available.

The need for a constraint set utility measure has become imperative with the recent observation that some constraint sets, even when completely accurate with respect to the evaluation labels, can actually decrease clustering performance [9]. The usual practice when describing the results of constrained clustering experiments is to report the clustering performance averaged over multiple trials, where each trial consists of a set of constraints that is randomly generated from the data labels. While it is generally the case that average performance does increase as more constraints are provided, a closer examination of the individual trials reveals that some, or even many, of them instead cause a drop in accuracy. Table 1 shows the results of 1000 trials, each with a different set of 25 randomly selected constraints, conducted over four UCI data sets [10] using four different k-means-based constrained clustering algorithms. The table reports the fraction of trials in which the performance was lower than the default k-means result, which ranges from 0% up to 87% of the trials.

The average performance numbers obscure this effect because the “good” trials tend to have a larger magnitude change in performance than the “bad” trials do. However, the fact that any of the constraint sets can cause a decrease in performance is unintuitive, and even worrisome, since the constraints are known to be noise-free and should not lead the algorithm astray.

To better understand the reasons for this effect, Davidson et al. [9] defined two constraint set properties and provided a quantitative way to measure them. *Informativeness* is the fraction of information in the constraint set that the algorithm cannot determine on its own. *Coherence* is the amount of agreement between the constraints in the set. Constraint sets with low coherence will be

Table 1. Fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy, compared to an unconstrained run with the same centroid initialization (table from Davidson et al. [9]).

Data Set	Algorithm			
	CKM [2] Constraint enforcement	PKM [5] Constraint enforcement	MKM [5] Metric learning	MPKM [5] Enforcement and metric learning
Glass	28%	1%	11%	0%
Ionosphere	26%	77%	0%	77%
Iris	29%	19%	36%	36%
Wine	38%	34%	87%	74%

difficult to completely satisfy and can lead the algorithm into unpromising areas of the search space. Both high informativeness and high coherence tend to result in an increase in clustering performance. However, these properties do not fully explain some clustering behavior. For example, a set of just three randomly selected constraints, with high informativeness and coherence, can increase clustering performance on the `iris` data set significantly, while a constraint set with similarly high values for both properties has no effect on the `ionosphere` data set. Additional work must be done to refine these measures or propose additional ones that better characterize the utility of the constraint set.

Two challenges for future progress in this area are: 1) to identify other constraint set properties that correlate with utility for constrained clustering algorithms, and 2) to learn to predict the overall utility of a new constraint set, based on extracted attributes such as these properties. It is likely that the latter will require the combination of several different constraint set properties, instead of being a single quantity, so using machine learning techniques to identify the mapping from properties to utility may be a useful approach.

2.2 Cost: How Can We Make Constraints Cheaper to Acquire?

A single pairwise constraint specifies a relationship between two data points. For a data set with n items, there are $\frac{1}{2}n(n-1)$ possible constraints. Therefore, the number of constraints needed to specify a given percentage of the relationships (say, 10%) increases quadratically with the data set size. For large data sets, the constraint specification effort can become a significant burden.

There are several ways to mitigate the cost of collecting constraints. If constraints are derived from a set of labeled items, we obtain $L(L-1)$ constraints for the cost of labeling only L items. If the constraints arise independently (not from labels), most constrained clustering algorithms can leverage constraint properties such as transitivity and entailment to deduce additional constraints automatically. A more efficient way to obtain the most useful constraints for the least effort is to permit the algorithm to actively solicit only the constraints it needs. Klein et al. [7] suggested an active constraint acquisition method in which a

hierarchical clustering algorithm can identify the m best queries to issue to the oracle. Recent work has also explored constraint acquisition methods for partitional clustering based on a farthest-first traversal scheme [11] or identifying points that are most likely to lie on cluster boundaries [12]. When constraints are derived from data labels, it is also possible to use an unsupervised support vector machine (SVM) to identify “pivot points” that are most useful to label [13].

A natural next step would be to combine methods for active constraint acquisition with methods for quantifying constraint set utility. In an ideal world, we would like to request the constraint(s) which will result in the largest increase in utility for the existing constraint set. Davidson et al. [9] showed that when restricting evaluation to the most coherent constraint sets, the average performance increased for most of the data sets studied. This early result suggests that coherence, and other utility measures, could be used to guide active constraint acquisition.

Challenges in this area are: 1) to incorporate measures of constraint set utility into an active constraint selection heuristic, akin to the MaxMin heuristic for classification [14], so that the best constraint can be identified and queried prior to knowing its designation (must/cannot), and 2) to identify efficient ways to query the user for constraint information at a higher level, such as a cluster description or heuristic rule that can be propagated down to individual items to produce a batch of constraints from a single user statement.

2.3 Sharing: When and How Should Constraints be Propagated to Neighboring Points?

Another way to get the most out of a set of constraints is to determine how they can be propagated to other nearby points. Existing methods that learn distance metrics use the constraints to “warp” the original distance metric to bring must-linked points closer together and to push cannot-linked points farther apart [7, 8, 6, 5]. They implicitly rely on the assumption that it is “safe” to propagate constraints locally, in feature space. For example, if a must be linked to b , and the distance $dist(a, c)$ is small, then when the distance metric is warped to bring a closer to b , it is also likely that the distance $dist(b, c)$ will shrink and the algorithm will cluster b and c together as well. The performance gains that have been achieved when adapting the distance metric to the constraints are a testament to the common reliability of this assumption.

However, the assumption that proximity can be used to propagate constraints is not always a valid one. It is only reasonable if the distance in feature space is consistent with the distances that are implied by the constraint set. This often holds true, since the features that are chosen to describe the data points are consistent with the data labels, which are commonly the source of the constraints. One exception is the `tic-tac-toe` data set from the UCI archive [10]. In this data set, each item is a 3x3 tic-tac-toe board that represents an end state for the game, assuming that the ‘x’ player played first. The boards are represented with nine features, one for each position on the board, and each one can take on a value of ‘x’, ‘o’, or ‘b’ (for blank). The goal is to separate the boards into two

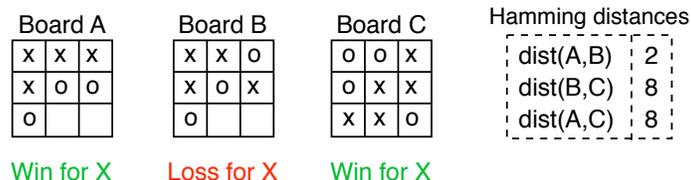


Fig. 1. Three items (endgame boards) from the tic-tac-toe data set. For clarity, blanks are represented as blanks, rather than spaces marked ‘b’. The Hamming distances between each pair of boards are shown on the right.

clusters: one with boards that show a win for ‘x’ and one with all other boards (losses and draws).

This data set is challenging because proximity in the feature space does not correlate well with similarity in terms of assigned labels. Consider the examples shown in Figure 1. Hamming distance is used with this data set, since the features have symbolic values. Boards A and B are very similar (Hamming distance of 2), but they should be joined by a cannot-link constraint. In contrast, boards A and C are very different (Hamming distance of 8), but they should be joined by a must-link constraint. In this situation, propagating constraints to nearby (similar) items will not help improve performance (and may even degrade it).

Clustering performance on this data set is typically poor, unless a large number of constraints are available. The basic k-means algorithm achieves a Rand Index of 51%; COP-KMEANS requires 500 randomly selected constraints to increase performance to 92% [2]. COP-COBWEB is unable to increase its performance above the baseline of 49% performance, regardless of the number of constraints provided [1]. In fact, when we examine performance on a held-out subset of the data¹, it only increases to 55% for COP-KMEANS, far lower than the 92% performance on the rest of the data set. For most data sets, the held-out performance is much higher [2]. The low held-out performance indicates that the algorithm is unable to generalize the constraint information beyond the exact items that participate in constraints. This is a sign that the constraints and the features are not consistent, and that propagating constraints may be dangerous. The results of applying metric learning methods to this data set have not yet been published, probably because the feature values are symbolic rather than real-valued. However, we expect that metric learning would be ineffective in this case.

Challenges to be addressed in this area are: 1) to characterize data sets in terms of whether or not constraints should be propagated (when is it “safe” and when should the data overrule the constraints?), and 2) to determine the degree to which the constraints should be propagated (e.g., how far should the

¹ The data subset is “held-out” in the sense that no constraints were generated on the subset, although it was clustered along with all of the other items once the constraints were introduced.

local neighborhood extend, for each constraint?). It is possible that constraint set coherence [9] could be used to help estimate the relevant neighborhood for each point.

3 Conclusions

This paper outlines several important unanswered questions that relate to the practice of constrained clustering. To use constrained clustering methods effectively, it is important that we have tools for estimating the *value* of a given constraint set prior to clustering. We also seek to minimize the *cost* of acquiring constraints. Finally, we require guidance in determining when and how to *share* or propagate constraints to their local neighborhoods. In addressing each of these subjects, we will make it possible to confidently apply constrained clustering methods to very large data sets in an efficient, principled fashion.

Acknowledgments. I would like to thank Sugato Basu and Ian Davidson for ongoing discussions on constrained clustering issues and their excellent tutorial, “Clustering with Constraints: Theory and Practice,” presented at KDD 2006. The research described in this paper was funded by the NSF ITR Program (grant #0325329) and was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the Seventeenth International Conference on Machine Learning. (2000) 1103–1110
2. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of the Eighteenth International Conference on Machine Learning. (2001) 577–584
3. Shental, N., Bar-Hillel, A., Hertz, T., Weinshall, D.: Computing Gaussian mixture models with EM using equivalence constraints. In: Advances in Neural Information Processing Systems 16. (2004)
4. Wagstaff, K.L.: Intelligent Clustering with Instance-Level Constraints. PhD thesis, Cornell University (2002)
5. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the Twenty-First International Conference on Machine Learning. (2004) 11–18
6. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research* **6** (2005) 937–965
7. Klein, D., Kamvar, S.D., Manning, C.D.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proceedings of the Nineteenth International Conference on Machine Learning. (2002) 307–313

8. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: *Advances in Neural Information Processing Systems 15*. (2003)
9. Davidson, I., Wagstaff, K.L., Basu, S.: Measuring constraint-set utility for partial clustering algorithms. In: *Proceedings of the Tenth European Conference on Principles and Practice of Knowledge Discovery in Databases*. (2006) 115–126
10. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html> (1998)
11. Basu, S., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: *Proceedings of the SIAM International Conference on Data Mining*. (2004) 333–344
12. Xu, Q., desJardins, M., Wagstaff, K.L.: Active constrained clustering by examining spectral eigenvectors. In: *Proceedings of the Eighth International Conference on Discovery Science*. (2005) 294–307
13. Xu, Q.: *Active Querying for Semi-supervised Clustering*. PhD thesis, University of Maryland, Baltimore County (2006)
14. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* **2** (2002) 45–66