A Survey of Clustering with Instance Level Constraints [1] [2]

Ian Davidson (davidson@cs.ucdavis.edu)
University of California - Davis
and
Sugato Basu (sugato@google.com)
Google Research

Clustering with constraints is an important recent development in the clustering literature. The addition of constraints allows users to incorporate domain expertise into the clustering process by explicitly specifying what are desirable properties in a clustering solution. This is particularly useful for applications in domains where considerable domain expertise already exists. In this first extensive survey of the field, we discuss the uses, benefits and importantly the problems associated with using constraints. We cover approaches that make use of constraints for partitional and hierarchical algorithms to both enforce the constraints or to learn a distance function from the constraints.

Categories and Subject Descriptors: I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*

General Terms: Data mining, clustering

Additional Key Words and Phrases: Data mining, clustering, semi-supervised learning, constraints

1. INTRODUCTION AND MOTIVATION

Semi-supervised learning is a recent innovation in data mining, machine learning and pattern recognition that is at the intersection of supervised and unsupervised learning. The field has received much attention since using both unlabeled data and some human supervision has many benefits, including allowing applications to fields where human annotating of each instance (such as labeling instances) is expensive. This new field can be broadly broken into two areas: a) the addition of unlabeled data to supervised classifier learning and b) the addition of some supervision into clustering. The former already has several excellent survey papers [Seeger 2000; Zhu 2005] that have served to quickly bring researchers up to speed in the area. However, though the area of adding supervision to clustering has been very popular in the data mining field (best paper awards in the area

[1]This work was presented as a tutorial at the IEEE ICDM Conference 2005 and ACM KDD Conference 2006. The slides of those tutorials are available at www.constrained-clustering.org
[2]Many of the algorithms presented in this paper are available for download at www.constrained-clustering.org

Author's addresses:
I. Davidson, Department of Computer Science, 1 Shields Avenue, 2063 Kemper Hall, Davis CA 95616.
S. Basu, Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043

were won for KDD 2004 [Basu et al. 2004], SDM 2005 [Davidson and Ravi 2005a] and IEEE 2005 [Gondek and Hofmann 2004]) no survey papers exist. Furthermore, relevant work on the topic is published in venues that may not be read by data mining researchers. The purpose of this survey paper is to span the relevant sub-fields of the area of clustering with supervision in the form of constraints and give a cohesive presentation of important algorithms and results. We hope this survey will prepare researchers to explore the field and practitioners to use existing algorithms.

Clustering is ubiquitously used in data mining as a method of discovering novel and actionable subsets within a set of data. Given a set of data X, the typical aim of partitional clustering is to form a k-block set partition Π_k of the data. The process of clustering is important since, being completely unsupervised, it allows the addition of structure to previously unstructured items such as free-form text documents. For example, [Cohn et al. 2003] discuss a problem faced by Yahoo!, namely that one is given very large corpora of text documents/papers/articles and asked to create a useful taxonomy so that similar documents are closer in the taxonomy. Once the taxonomy is formed, the documents can be efficiently browsed and accessed. Unconstrained clustering is ideal for this initial situation, since in this case little domain expertise exists to begin with. However, as data mining progresses into more demanding areas, the chance of finding actionable patterns consistent with background knowledge and expectation is limited. For example, [Wagstaff et al. 2001a] show that clustering GPS trace data from automobiles with k-means so as to find lanes produces clusters quite different from the required elongated clusters (see Figure 12). They instead make use of background knowledge that highway lanes are four meters in width, and any two cars more than four meters apart in the direction perpendicular to the road direction must be in different lanes/clusters. In this way, they are placing **constraints** on certain properties of the final clustering.

Clustering with constraints allows the incorporation of background domain expertise with work so far incorporating knowledge in the form of instance level constraints. The two types of constraints introduced by Wagstaff [Wagstaff and Cardie 2000] are must-link denoted by $c_{=}(x,y)$ and cannot-link denoted by $c_{\neq}(x,y)$, meaning that two instance x and y must be in the same cluster or cannot be in the same cluster respectively. Must-link and cannot-link constraints, though apparently simple, share interesting properties. Must-link constraints are an example of an equivalence relation and hence are symmetrical, reflexive and transitive; this means that $c_{=}(x,y)$ and $c_{=}(y,z) \Rightarrow c_{=}(x,z)$ such that x,y,z form a connected component, i.e., each is connected to the other via an explicit or implied must-link constraint. Formally:

Observation 1 ***Must-link Constraints are Transitive.*** *Let CC_i and CC_j be connected components (completely connected subgraphs by must-link constraints), and let x and y be the instances in CC_i and CC_j respectively. Then $c_{=}(x,y), x \in CC_i, y \in CC_j \Rightarrow c_{=}(a,b), \forall a,b : a \in CC_i, b \in CC_j$.*

Similarly connected components of must-link constraints can give rise to entailed cannot-link constraints.

Observation 2 ***Cannot-link Constraints Can Be Entailed.*** *Let CC_i and CC_j be connected components (completely connected subgraphs by must-link constraints), and let*

x and y be the instances in CC_i and CC_j respectively. Then $c_{\neq}(x,y), x \in CC_i, y \in CC_j \Rightarrow c_{\neq}(a,b), \forall a,b : a \in CC_i, b \in CC_j$.

Though apparently simple, must-link and cannot-link constraints are powerful. In sufficient numbers they can shatter the training set X and specify any set partition of X. Furthermore, these two constraints can specify interesting spatial properties [Davidson and Ravi 2005a] [Davidson and Ravi 2007]. If we wish two clusters to have their points separated by δ or greater distance then this is equivalent to clustering with a conjunction of must-links between all instances that are **closer** than δ distance together. Similarly, if we wish the cluster diameters to be at most α then this is equivalent to placing a conjunction of cannot-link constraints between all instances greater than α distance apart. Finally, if we wish each point in a cluster to have a neighbor within ε distance then this can be represented using a disjunction of ML constraints. Details for some geometric constraints are shown in Figure 1.

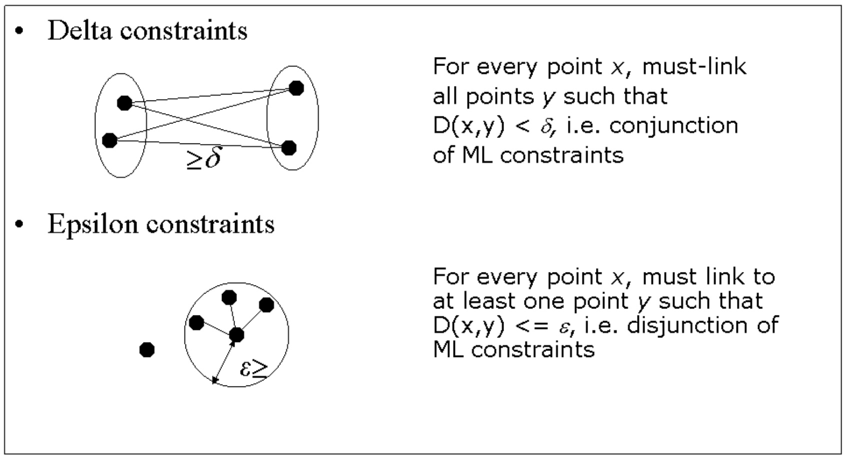

• Delta constraints
$\geq\delta$
For every point x, must-link all points y such that D(x,y) < δ, i.e. conjunction of ML constraints
• Epsilon constraints
$\varepsilon\geq$
For every point x, must link to at least one point y such that D(x,y) <= ε, i.e. disjunction of ML constraints

Fig. 1. Delta and Epsilon Constraints

We begin this survey by describing, at a high level, how constraints can be used in clustering in Section 2. In Section 3 we describe several successful applications and how constraints were generated in specific domains. The next section, Section 4, summarizes reported benefits and problems associated with using constraints. We believe this is particularly important, since the problems associated with using constraints are not be well discussed in the literature. We next give an outline of specific algorithms that make use of instance level constraints. In Section 5, we discuss algorithms for partitional (non-hierarchical) clustering that attempt to satisfy all or most constraints, with the latter allowing for ignoring noisy or poorly specified constraints. Algorithms for initializing and generating constraints are outlined in Section 6, while hierarchical clustering that attempt to learn a distance matrix from the constraints or satisfy them all are discussed in Section 7. Next, Section 8 discusses the general problem of learning a distance *function* (not a instance-wise distance matrix) from all the constraints and then its subsequent use in clustering. Section 9 discusses the dual objective problem of learning a distance function while allowing constraints to be ignored.

Throughout this paper we use the notation described in section 11.

2. USES OF CONSTRAINTS

The typical supervised learning situations involves having a label associated with each instance. The semi-supervised learning situation is when only a small subset of instances have labels. If the available labeled data represent all the relevant categories, then semi-supervised classification algorithms can be readily used for data categorization. For details see the various algorithm surveys [Seeger 2000; Zhu 2005]. However in many domains, knowledge of the relevant categories is incomplete and pairwise constraints are often a more readily available form of supervision than labels in certain clustering tasks. Examples of such domains are given in Section 3. Moreover, in an interactive learning setting, a user who is not a domain expert can sometimes provide feedback in the form of must-link and cannot-link constraints [Cohn et al. 2003; Davidson et al. 2007] more easily than class labels, since providing constraints does not require the user to have significant prior knowledge about the categories in the data set or even for such categories to exist.

Constraints have typically been used in clustering algorithms in two ways. They can be used to modify the cluster assignment stage of the cluster algorithm so as to enforce satisfaction of the constraints as much as possible. Alternatively, the distance function of the clustering algorithm can also be trained from the constraints either before or during the actual clustering. In all of these cases, constraints can also be used in the initialization phase, where the initial clusters are formed such that must-linked instances are in the same clusters and cannot-linked instances are in different clusters. Based on this categorization, existing methods for constrained clustering can be put into two general approaches that we call *constraint-based* and *distance-based* methods.

2.1 Constraint-based methods

In constraint-based approaches, the clustering algorithm itself is modified so that the available constraints are used to bias the search for an appropriate clustering of the data. The pairwise constraints specify whether two instances should be in the same cluster (must-link) or in different clusters (cannot-link).

There have typically been two types of constraint-based approaches: (1) ones with strict enforcement, which find the best feasible clustering respecting all the given constraints [Wagstaff et al. 2001b; Davidson and Ravi 2005b], and (2) ones with partial enforcement, which find the best clustering while maximally respecting constraints [Basu et al. 2004; Segal et al. 2003; Davidson and Ravi 2005a; Law et al. 2005]. Figure 3 shows an example of a clustering which respects all the given constraints in Figure 2. Details of these algorithms are outlined in section 5 for partitional algorithms and section 7 for hierarchical algorithms.

Constraint-based clustering has been done using several techniques:

—modifying the clustering objective function so that it includes a term (penalty) for satisfying specified constraints [Demiriz et al. 1999; Davidson and Ravi 2005a].

—clustering using side-information from conditional distributions in an auxiliary space [Sinkkonen and Kaski 2000]

—enforcing all constraints to be satisfied during the assignment step in the clustering process [Wagstaff et al. 2001b]

—initializing clusters and inferring clustering constraints based on neighborhoods derived from labeled examples [Basu et al. 2002].

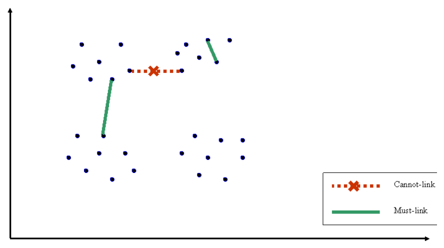


Fig. 2. Input instances and constraints

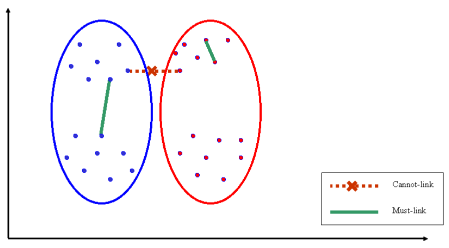


Fig. 3. A Clustering That Satisfies All Constraints

2.2 Distance-based methods

In distance-based approaches, an existing clustering algorithm that uses a distance measure is employed. However, rather than use the Euclidean distance metric, the distance measure is first trained to "satisfy" the given constraints. In this context, satisfying the constraints means that must-linked (similar) instances are close together and cannot-linked (different) instances are far apart in the learnt distance space. Several distance measures have been used for distance-based constrained clustering:

—string-edit distance trained using EM [Bilenko and Mooney 2003],

—Jensen-Shannon divergence trained using gradient descent [Cohn et al. 2003],

—Euclidean distance modified by a shortest-path algorithm [Klein et al. 2002] and

—Mahalanobis distances trained using convex optimization [Bar-Hillel et al. 2003; Xing et al. 2003]

Several clustering algorithms using trained distance measures have been employed for constrained clustering, including single-link [Bilenko and Mooney 2003] and complete-link [Klein et al. 2002] agglomerative clustering, EM [Cohn et al. 2003; Bar-Hillel et al. 2003], and KMeans [Bar-Hillel et al. 2003; Xing et al. 2003]. Recent techniques in distance-metric learning for clustering include learning a margin-based clustering distortion measure using boosting [Hertz et al. 2004], and learning a distance metric transformation that is globally linear but locally non-linear [Chang and Yeung 2004]. Figure 5 shows a simple example of learning a distance function from the constraints given in Figure 4 and then clustering. Notice that in Figure 5 the input data space has been stretched in the horizontal dimension and compressed in the vertical dimension, to draw the must-linked instances closer and put the cannot-linked instances farther apart. Section 8 outlines methods of learning distance functions from constraints.

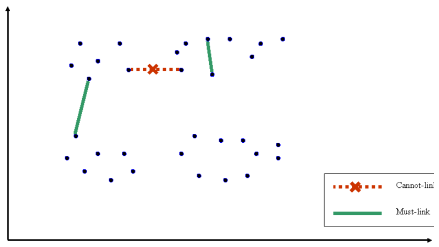


Fig. 4. Input instances and Constraints

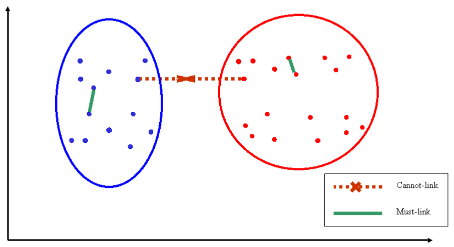


Fig. 5. A Learnt distance space respective of the constraints.

There have been some algorithms that try to both enforce constraints and learn distance functions from constraints — details of these algorithms will be presented in section 9.

3. EXAMPLE APPLICATIONS OF USING CONSTRAINTS

This section outlines some real-world examples where constraints are used in clustering. For each we discuss how the constraints are generated and how they improved the resultant clusterings.

3.1 Image data

Figures 6 shows a dataset where the task is to cluster faces from the CMU Faces Database. The method used here is the one which is most popular in the literature for generating constraints — set the number of clusters to the number of classes in the dataset, and then generate constraints from the labels: if two instances have the same class label, then add a must-link between them, and if two instances have different class labels, then add a cannot-link between them. In this case, the goal is to cluster face images according to their orientation, as shown in Figure 6. So, the two images which are selected in Figure 7 have a cannot-link between them, even though they are of the same person, since they have different facial orientation (one is in the class “UP” and the other in the class “SIDE”).

Another image dataset where constraints are used for clustering is shown in Figure 8. Here the task is to cluster pixels in an image into segments with the goal of performing object identification for Aibo robot navigation [Davidson and Ravi 2005a]. In this case, more complex cluster-level/spatial constraints are used (e.g., ε and δ constraints described in Figure 1 [Davidson and Ravi 2005a]) to aid in creating clusters that are well seperated and hence useful for the purpose of creating a path for robot navigation.

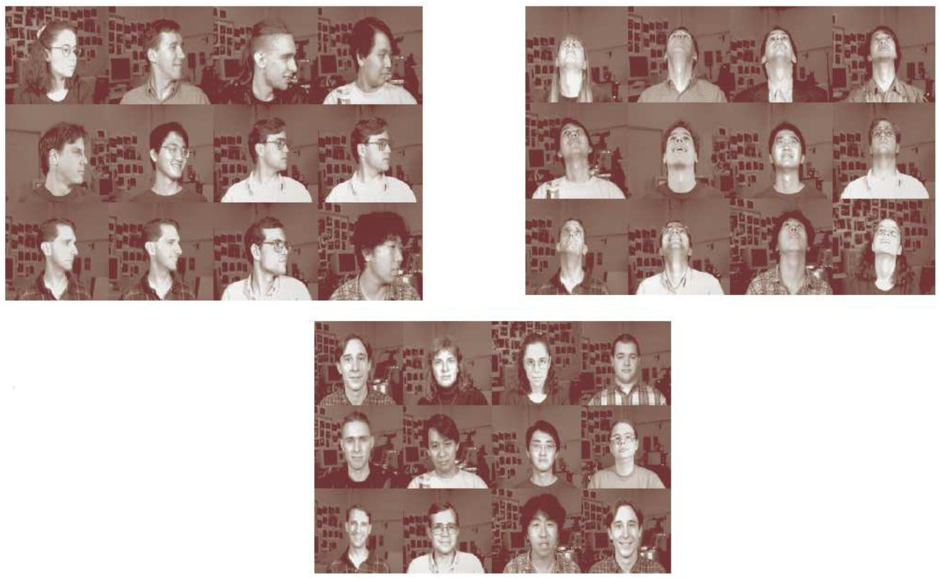


Fig. 6. CMU Faces Dataset

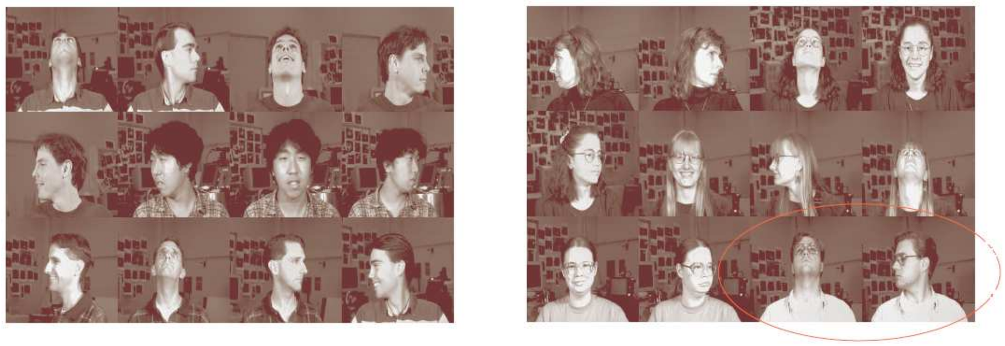


Fig. 7. Cannot-link constraint between different face orientation

3.2 Video data

Video data is an example where constraints can be generated readily from the domain, especially from spatio-temporal aspects of video sequences [Yan et al. 2004]. In temporally successive frames of video, one can add must-links between groups of pixels that represent the same object, when the task is to perform object recognition by segmentation and clustering. It is also possible to add cannot-link constraints between two different image segments of the same video snapshot, since they have a low chance of belonging to the same object after segmentation is performed. An example taken from [Yan et al. 2004] is shown in Figure 9. In fact, in this domain, there is such a plethora of constraints that active learning techniques are used to select the most useful constraints [Yan et al.

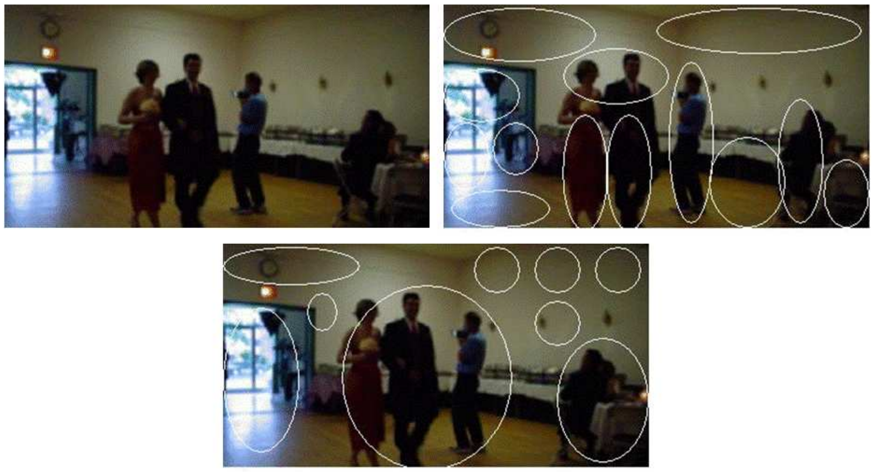


Fig. 8. Spatial clusters for use in Aibo robot navigation. The three pictures indicate (a) the original image, (b) clusters found by unconstrained *k*-means, (c) improved clusters using cluster-level spatial constraints

2004]. An interesting question in this context is what happens if too many constraints are created and used in the constrained clustering algorithm — does this make the problem over-constrained? Section 4 discusses this and related questions regarding the practical issues and difficulties of using constraints in clustering.

3.3 Biological data

In gene clustering based on micro-array data, genes are represented using their expression profile in different experiments and clustered using different algorithms, e.g., hierarchical clustering [Eisen et al. 1998]. An example is shown in Figure 10. In this case, must-link constraints can be generated between genes using co-occurrence data from the Database of Interacting Proteins, which contains information about which genes (and corresponding proteins) co-occur in cellular processes [Xenarios et al. 2001]. These constraints can then be used to improve the clustering process [Segal et al. 2003].

3.4 Text data

In content-management tasks (routinely performed by companies like Google, Interwoven or Verity), the goal is to automatically categorize large amounts (often in the order of millions) of text documents into groups or clusters. In this case, constraints can be obtained from multiple auxiliary sources, e.g., the co-occurrence of two documents in a directory can be used to infer a must-link constraint between the documents, two documents in

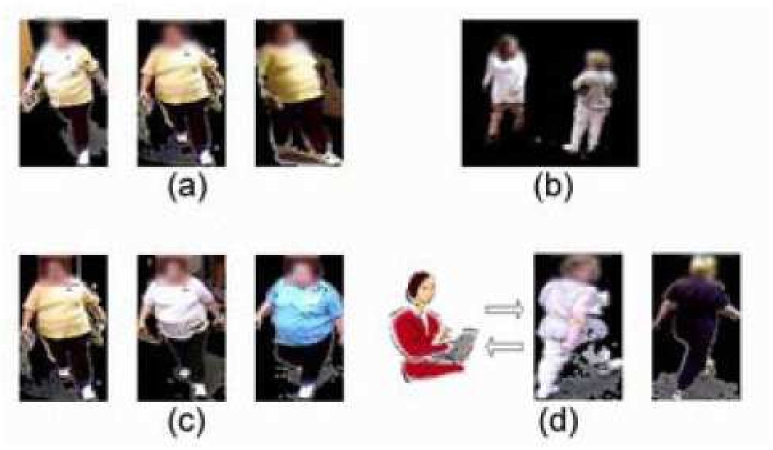


Fig. 9. Different kinds of pairwise constraints: (a) Temporal constraints from single tracking sequence of a person (b) Temporal constraints of different regions extracted at the same time (c) Constraints provided by comparing faces (d) Constraints provided by user feedback.

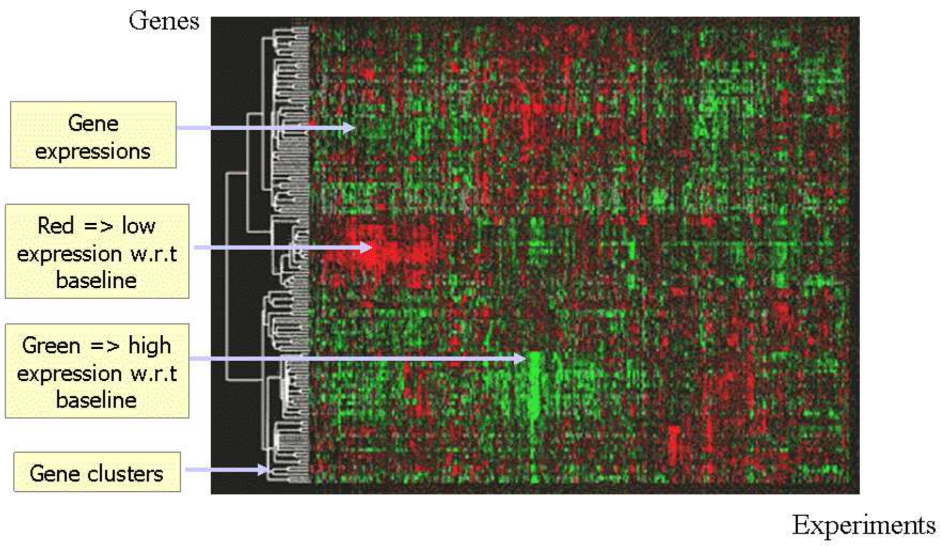


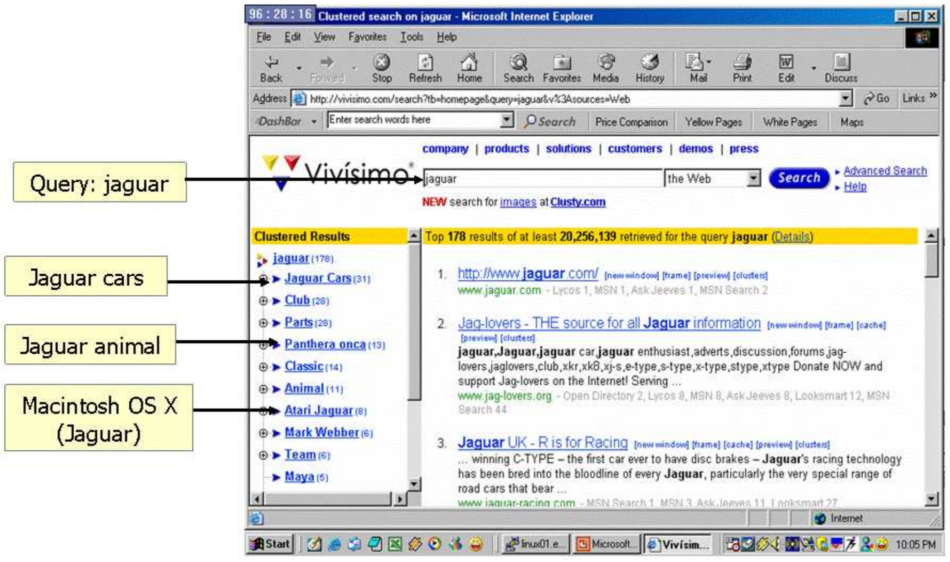
Fig. 10. Clustering of gene microarray data

different categories of the Open Directory Project[3] hierarchy can be considered as cannot-linked, etc. Using these constraints from the auxiliary data sources, one can customize the clustering output for the particular task, e.g., make a document hierarchy that is close to the input directory structure in which the documents are placed.

[3]www.dmoz.org

3.5 Web data

Constrained clustering is quite useful in post processing search results, as performed by companies like Vivisimo.[4] Here, the goal is to automatically cluster the results of ambiguous search-engine queries like “jaguar” into clusters of URLs that refer to concepts like “Jaguar cars”, “Jaguar animal” or “Jaguar Mac OS” (as shown in Figure 11). In this case, constraints can be mined from query sessions in web logs – one can get valuable information regarding which websites are visited together, by analyzing co-occurrence of url’s within the same user session. Clustering using this auxiliary data can help in biasing the search result clustering towards the preferences of the user.


Query: jaguar
Jaguar cars
Jaguar animal
Macintosh OS X (Jaguar)
96 : 28 : 16 Clustered search on jaguar - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://vivisimo.com/search?tb=homepage&query=jaguar&v%3Asources=Web
Go Links
DashBar Enter search words here Search Price Comparison Yellow Pages White Pages Maps
company | products | solutions | customers | demos | press
Vivísimo
jaguar
the Web
Search
Advanced Search
Help
NEW search for images at Clusty.com
Clustered Results
jaguar (178)
Jaguar Cars (31)
Club (28)
Parts (28)
Panthera onca (13)
Classic (14)
Animal (11)
Atari Jaguar (8)
Mark Webber (6)
Team (6)
Maya (5)
Internet
Start
linux01.e...
Microsoft...
Vivísim...
10:05 PM

Top 178 results of at least 20,256,139 retrieved for the query jaguar (Details)

1. http://www.jaguar.com/ [new window] [frame] [preview] [clusters]
www.jaguar.com - Lycos 1, MSN 1, Ask Jeeves 1, MSN Search 2

2. Jag-lovers - THE source for all Jaguar information [new window] [frame] [cache] [preview] [clusters]
jaguar,Jaguar,jaguar car,jaguar enthusiast,adverts,discussion,forums,jag-lovers,jaglovers,club,xkr,xk8,xj-s,e-type,s-type,x-type,stype,xtype Donate NOW and support Jag-lovers on the Internet! Serving ...
www.jag-lovers.org - Open Directory 2, Lycos 8, MSN 8, Ask Jeeves 8, Looksmart 12, MSN Search 44

3. Jaguar UK - R is for Racing [new window] [frame] [cache] [preview] [clusters]
... winning C-TYPE – the first car ever to have disc brakes – Jaguar's racing technology has been bred into the bloodline of every Jaguar, particularly the very special range of road cars that bear ...
www.jaguar-racing.com - MSN Search 1, MSN 3, Ask Jeeves 11, Looksmart 27

Fig. 11. Clustering of web search results

3.6 Audio data

In certain audio analysis tasks, complete class labels may be unknown but constraints can be obtained directly from the domain. This occurs in the context of clustering for speaker identification in a conversation [Bar-Hillel et al. 2003] — it is often not possible to know the number of speakers upfront in a dataset, but it is easier to detect if two speakers are similar or different. In such a case, constrained clustering is a more natural analysis framework than classification-based algorithms.

3.7 GPS data

Constrained clustering of GPS data is used for lane-finding, as shown in Figure 12 [Wagstaff et al. 2001b]. Data instances are represented by the (x,y) location of a car on the road as recorded from GPS traces; it is also known when a car changes lanes, but not which lane it

[4]www.vivisimo.com

changes to. Figure 13 shows an example such a GPS trace where each instance represents a car and multiple instances can refer to the same car at different points in time.

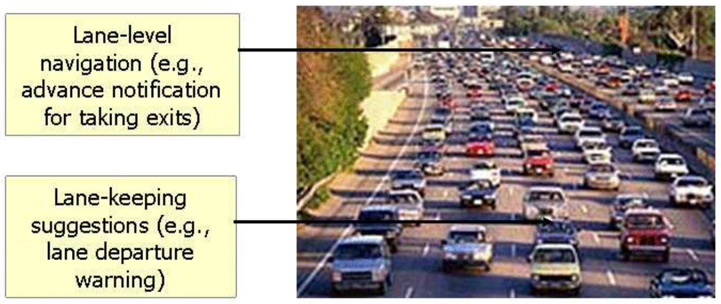

Lane-level navigation (e.g., advance notification for taking exits)
Lane-keeping suggestions (e.g., lane departure warning)

Fig. 12. Uses of GPS trace information.

In this domain, true clusters are very elongated and horizontally aligned with the lane central lines. Constraints in this case include must-link constraints inferred from trace-contiguity (that is, must-links between the same cars at different positions in the trace) and cannot-link constraints inferred from cars that are more than 4 meters apart in a direction perpendicular to the car travel. The later is used since the maximum width of a lane is 4 meters, hence cars more than this distance apart must be in different lanes. These constraints have been shown to be very useful for finding clusters in the data [Wagstaff et al. 2001b], which are useful for different tasks like lane-level navigation (e.g., advance notification for taking exits) and lane-keeping suggestions (e.g., lane departure warnings).

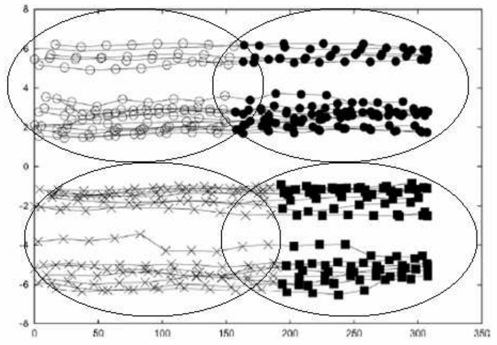


Fig. 13. Clusters found in GPS data for lane finding without using constraints.

4. BENEFITS AND PROBLEMS OF USING CONSTRAINTS

In this section we sketch and reference the well discussed benefits of using constraints and spend more time discussing the less discussed problems.

4.1 Benefits

There are two main reported benefits in the literature to using constraints: a) Improved accuracy at predicting extrinsic labels for all instances when generating constraints from a few labeled data and b) Creating clusters with desirable geometric properties. We now discuss these two benefits.

Given a large set of unlabeled data $X = \{x_1 \dots x_u\}$ and a small set of labeled data $L = \{(x_{u+1}, y_{u+1}) \dots (x_{u+l}, y_{u+l})\}$ it is common to randomly with replacement choose two instances from L. If the two instances' labels agree (disagree), then generate a must-link (cannot-link) constraint between them. A typical method of measuring the performance of a clustering algorithm is its accuracy on predicting extrinsic labels not given to it, in this case the labels for X. This typically requires that the clustering algorithm use $k = k^*$ where k^* is the number of different extrinsic labels. Measures such as the Rand index [Rand 1971], normalized mutual information [Basu et al. 2004] and just the highest proportion of instances with the same label in a cluster weighted by cluster size are common measures. The seminal work by Wagstaff and Cardie [Wagstaff and Cardie 2000] which generated constraints from a small set of the labeled data illustrated convincingly that constraints improved clustering accuracy. For the UCI Soybean, Pos and Mushroom, data sets as little as 40 constraints produced accuracy increases over 6%, 10% and 20% respectively with more constraints producing even better improvements. Wagstaff and collaborators experiments reported the **average** accuracy improvement over many constraint sets that we shall see masks interesting results. A well established relationship that as the number of constraints increases the accuracy increases has been shown extensively by many researchers [Wagstaff et al. 2001a; Klein et al. 2002; Xing et al. 2003; Basu et al. 2004; Bar-Hillel et al. 2005; Wagstaff 2002; Lu and Leen 2005; Davidson and Ravi 2005a].

Observation 3 ***Constraints Improve Average Case Accuracy.*** *The performance of predicting an extrinsic label when averaged over many many different constraint sets will typically increase over not using any constraints.*

Unusually, for the Tic-Tac-Toe dataset, for no amount of constraints produced an increase in cluster accuracy beyond a few percentage points, at predicting the extrinsic labels. A plausible explanation given by the authors is that using $k = k^*$ is not appropriate.

Xing and collaborators [Xing et al. 2003] show that when learning a distance function for nine UCI data sets, even a small amount of constraints can improve performance. Their results also show that combining metric learning and constraint satisfaction (for just ML constraints) provides better performance than just learning a distance metric. In all nine data sets (which did not contain Tic-Tac-Toe) learning a distance metric from even a small number of constraints produced improved accuracy.

A lesser discussed benefit is the ability to use constraints to create clusters of desirable shape. For example, Wagstaff et al. [Wagstaff et al. 2001a] illustrate that clustering GPS trace data from automobiles using the k-means algorithm produces clusters which are quite different from the elongated clusters (representing lanes) that one would expect. However, when clustering with cannot-link constraints between all instances greater than four meters apart in the direction perpendicular to the road the resulting clusters have the desired shape. The use of instance level constraints as geometric constraints is further discussed in [Davidson and Ravi 2005a; 2007]. A key insight of that work is that geometric properties can be represented by combinations of many instance level constraints. As Figure 1 shows, a minimum separation constraint that any two instances in two clusters be at least δ distance apart can be represented as a conjunction of ML constraints between all instances less than or equal to δ distance apart. Similarly, if a point must have an ϵ close-by neighbor is equivalent to a disjunction of ML constraints. Further work along these lines will allow specifying user-preferences in a more intuitive way than instance level constraints.

4.2 Problems

Though there have been much reported success with using constraints and clustering, there are two major limitations that are typically not discussed in the literature. Hence we devote more space discussing them in detail.

4.2.1 *Feasibility.* The introduction of constraints into clustering changes the clustering problem to be: *Find the best clustering that satisfies all constraints.* However, if the constraints are poorly specified they may directly or indirectly contradict each other in such a way that **no assignment of instances to clusters satisfies all constraints**. For example, there are clearly no clusterings for the constraints *ML*(x_1,x_2), *CL*(x_1,x_2) regardless of the value of k. Similarly for $k = 2$ and the constraints *CL*(x_1,x_2), *CL*(x_2,x_3), *CL*(x_1,x_3) there exists no feasible clustering. Formally for non-hierarchical clustering the *feasibility* problem is defined as:

Definition 1 ***(Feasibility Problem [Davidson and Ravi 2005a])*** *Given a data set X, collection of constraints C, a lower bound K_ℓ and an upper bound K_u on the number of clusters, does there exist a partition of X into k blocks such that $K_\ell \leq k \leq K_u$ and all the constraints in C are satisfied?*

The complexity results for constraints in various combination are in Tables I and II. As we can see using CL constraints makes the feasibibility problem intractable and hence clustering with constraints intractable. This is so since if finding a single clustering that satisfies all constraints is difficult by necessity so will finding the best clustering that satisfies all constraints. The remainder of this section discusses feasibility for non-hierarchical clustering. The feasibility problem for hierarchical clustering is discussed in section 7.

Observation 4 ***Knowing a Feasible Solution Exists, Does Not Help Us Find It.*** *It should be noted that the implications of these complexity result is that even if there is a feasible solution it* ***does not mean*** *it will be easy to find.*

Both Wagstaff [Wagstaff 2002] and Davidson and Ravi [Davidson and Ravi 2007] show that even with clustering with $k = k^*$ (hence guaranteeing a feasible solution) simple algorithms such as COP-k-means (see section 5) will not converge due to the feasibility problem being intractable. Examples showing this phenomenon are in Figure 4.2.1 and indicate that adding CL constraints can quickly over-constraint the clustering under constraints problem so that satisfying all constraints is difficult [Davidson and Ravi 2006].

4.2.2 *Not All Constraint Sets are Useful.* The assumption made by constrained clustering algorithms is that the constraints are hints that help guide the algorithm to the desired partition. On this premise the more information (constraints) given the greater the agreement between the output partition and the desired partition should be. This has been shown to be the case *on average* (see Table III).

Constraint	Conjunction	CNF version	DNF version	Choice Set
Must-Link	**P**	**NP**-complete	**P**	**P**
Cannot-Link	**NP**-Complete	**NP**-complete	**NP**-Complete	**P**

Table I. Complexity of Feasibility Problem for Instance-Level Constraints

Constraint	Complexity
δ-constraint	**P**
ε-constraint	**P**
Must-Link and δ	**P**
Must-Link and ε	**NP**-complete
δ and ε	**P**
Cannot-Link and any other constraint	**NP**-complete

Table II. Complexity of Feasibility Problems for Cluster-Level and Combinations of Constraints

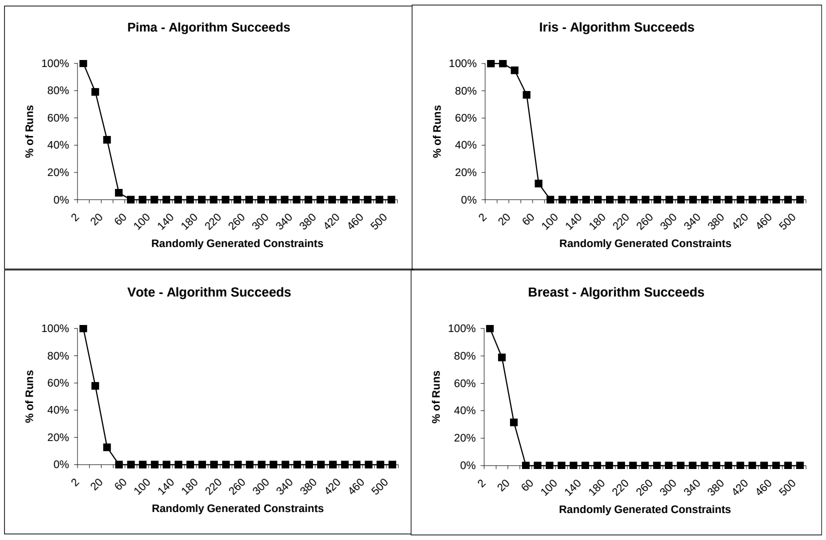


Fig. 14. Graph of the proportion of times from 500 independent trials the algorithm in Figure 17 converges (y-axis) for various numbers of randomly chosen CL only constraints (x-axis).

Despite observation 3, Davidson, Wagstaff and Basu [Davidson et al. 2006] show that even if the constraints are noise-free and generated from the ground truth, then it is possible for **individual** constraint sets to **decrease** clustering accuracy. This appears to disagree with experimental results produced by others. This is so since the experimental methodology adopted by others involves **averaging** the constrained clustering algorithm performance over many constraint sets. The resultant learning curves are produced by repeating this process for different constraint set sizes, and the typical result is that, on average, when

Table III. Average performance (Rand Index) of four constrained clustering algorithms, for 1000 trials with 25 randomly selected constraints. The best result for each algorithm/data set combination is in bold. These results confirm observation 3

	Algorithm							
	CKM		**PKM**		**MKM**		**MPKM**	
Data	None	Const.	None	Const.	None	Const.	None	Const.
Glass	69.0	**69.4**	43.4	**68.8**	39.5	**56.6**	39.5	**67.8**
Ionosphere	58.6	**58.7**	58.8	**58.9**	**58.9**	**58.9**	**58.9**	**58.9**
Iris	84.7	**87.8**	84.3	**88.3**	88.0	**93.6**	88.0	**91.8**
Wine	70.2	**70.9**	71.7	**72.0**	**93.3**	91.3	**93.3**	90.6

more constraints are provided, clustering accuracy increases [Wagstaff et al. 2001a; Klein et al. 2002; Xing et al. 2003; Basu et al. 2004; Bar-Hillel et al. 2005; Wagstaff 2002; Lu and Leen 2005; Davidson and Ravi 2005a].

Observation 5 ***Individual Constraint Sets Can Have Adverse Effects.*** *Some constraint sets generated from the same ground truth set of labels that will evaluate the clusters may decrease the accuracy at predicting those very labels.*

Davidson, Wagstaff and Basu explore four different constrained clustering algorithms on several standard clustering data sets. The four algorithms represent two major types of constrained clustering techniques that attempt to satisfy most or all constraints (see section 5) or learn a distance metric (see section 8). The four algorithms used are described below:

—COP-KMeans (CKM) performs hard constraint satisfaction [Wagstaff et al. 2001a].

—PC-KMeans (PKM) performs soft constraint satisfaction (permits some constraints to be violated) [Basu et al. 2004].

—M-KMeans (MKM) performs metric learning from constraints, but does not require that the constraints be satisfied [Basu et al. 2004].

—MPC-KMeans (MPKM) is a hybrid approach, performing both soft constraint satisfaction and metric learning [Basu et al. 2004].

Table III shows the results (averaged over 1000 constraint sets) for each algorithm for both unconstrained and constrained performance when provided with 25 randomly selected constraints. We evaluated these algorithms on four UCI data sets [Blake and Merz 1998]: Glass ($n = 214$), Ionosphere ($n = 351$), Iris ($n = 150$), and Wine ($n = 178$). Clustering performance was measured in terms of the Rand Index [Rand 1971]. As expected [Wagstaff et al. 2001a; Xing et al. 2003; Basu et al. 2004], the average constrained clustering accuracy was typically greater than the average unconstrained accuracy. The exception is MKM and MPKM's performance on the Wine data set.

However, looking into each of these 1000 constraint sets we see that in Table IV that the fraction of these 1000 trials that suffered **a drop in clustering accuracy** when using constraints is not insignificant. Since each trial involved the same initialization of the centroids for both the unconstrained and constraint experiments performance differences are due to the use of constraints. These negative results occur frequently for many data sets and algorithms and naturally motivate the question of what properties occur in useful

Table IV. Fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy, compared to an unconstrained run with the same centroid initialization.

	Algorithm			
Data	**CKM**	**PKM**	**MKM**	**MPKM**
Glass	28%	1%	11%	0%
Ionosphere	26%	77%	0%	77%
Iris	29%	19%	36%	36%
Wine	38%	34%	87%	74%

constraint sets. Davidson, Wagstaff and Basu in their work discuss two metrics that can be used to identify useful constraint sets that improve clustering accuracy.

4.3 Problem Work Arounds

4.3.1 *Identifying Easy To Satisfy Sets of Constraints.* Prior to 2006, little attention was given to understanding when clustering under constraint problems became over-constrained and what constitutes a useful set of constraints. Wagstaff [Wagstaff 2002] and Davidson & Ravi [Davidson and Ravi 2005a] had earlier published sharp transition type effects when running algorithms that attempt to satisfy all constraints (i.e. COP-*k*-means). For increasing numbers of randomly generated constraints from labeled data the problem quickly becomes over-constrained even with many random restarts of the algorithms (see Figure 4.2.1). In earlier work [Davidson and Ravi 2006] the over-constraining phenomenon due to clustering under CL constraints was shown to be analogous to graph coloring and that COP-*k*-means is effectively a greedy coloring algorithm.

Observation 6 ***Constrained Clustering and Graph Coloring.***
Clustering to Satisfy CL constraints involves solving the graph coloring problem.

This result allows many results in graph coloring to be applicable to clustering to satisfy all constraints. For example, Brooks's theorem states that coloring is easy (tractable) when the number of colors (*k* in our situation) is greater than the maximum degree of the graph. This is precisely the situation that occurs when COP-*k*-means always converges in experimental results[Davidson and Ravi 2006].

Observation 7 ***Brooks's Result For Constrained Clustering.***
If $k > (Most\ CL\ Constraints\ On\ One\ Instance)$ *then there will always be a feasible clustering.*

Furthermore, though observation 4 states that in general finding a feasible solution is difficult when constraint sets meet these sufficient conditions we can always generate feasible solutions in polynomial time. For example, rather than sampling instance at random to generate constraints, to ensure Brooks's condition make sure that one instance is not part of more than *k* CL constraints. The issue of how to generate easy constraint sets is further discussed in [Davidson and Ravi 2006].

4.3.2 *Identifying Useful Sets of Constraints.* Davidson, Wagstaff and Basu created two measures for a constraint set: *informativeness* and *coherence* to identify useful constraint sets.

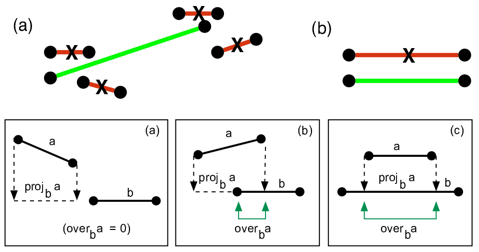


Fig. 15. Top: Illustrative examples of (a) informative constraints for COP-*k*-means and (b) incoherent constraints, for a Euclidean metric. ML constraints are solid lines; CL constraints contain an 'X'. Bottom: Three cases of computing coherence by projected overlap of constraints *a* and *b*.

Informativeness refers to the amount of information in the constraint set that the algorithm cannot determine on its own. For example, in Figure 15(a) (top-left), an algorithm such as COP-*k*-means would be biased towards grouping nearby instances together and separating distant instances, but the specified constraints contradict this bias. The informativeness is estimated by using the constraints as a hold-out/test set and measuring the algorithm's ability to predict them. Given a set of constraints C and an algorithm $\mathcal{A}$, we generate the best (lowest objective function) partition $P_{\mathcal{A}}$ by running $\mathcal{A}$ on the data set *without* any constraints. The fraction of constraints in C that are unsatisfied by $P_{\mathcal{A}}$ is then calculated:

$$I_{\mathcal{A}}(C) = \frac{1}{|C|}\left[\sum_{c\in C} unsat(c, P_{\mathcal{A}})\right] \tag{1}$$

Coherence measures the amount of agreement within the constraints themselves, with respect to a given distance metric ($\mathcal{D}$). Figure 15(b) (top-right) shows two constraints (ML and CL) that are very close and parallel but this is a contradiction since a ML (CL) constraints indicate that the distance between the instances and surrounding instances should be small (far). The amount of agreement is measured by the amount of overlap between constraints when represented as vectors as shown in Figure 15 (bottom). The authors results [Davidson et al. 2006] shows that highly informative and coherent constraint sets almost always increased a wide variety of constrained clustering algorithm's accuracy at predicting the labels the constraints were generated from as measured by the Rand index [Rand 1971].

5. PARTITIONAL CLUSTERING ALGORITHMS

The very first algorithms that made use of constraints were variations of the popular *k*-means iterative algorithm that attempt to find a set partition of the data instances. Recall that the purpose of the *k*-means algorithm is to find the set partition that (locally) minimizes the vector quantization error (also known as the distortion) shown in equation 2. As mentioned earlier, we use the notation in section 11.

$$VQE \;=\; \sum_{j=1}^{k} VQE_j \tag{2}$$

Input: A data set $X = \{x_1 \dots x_n\}$ to cluster, k: the number of clusters to find.
Output: A partition of $X, \Pi_k = \{\pi_1 \cup \pi_k\}$ into k clusters that is a local optima of the VQE (equation 2).

```
(1)  Randomly generate cluster centroids $\mu_1, \dots, \mu_k$.
(2)  loop until convergence do
     .  (a) for $i = 1$ to $|X|$ do
     .      (a.1) Assign $x_i$ to the nearest, in terms of distance to centroids, cluster.
     .  (b) Recalculate centroids $\mu_1, \dots, \mu_k$ according to equation 5.
(3)  Return with success.
```

Fig. 16. Clustering Using *k*-means

$$VQE_j = \frac{1}{2} \sum_{x_i \in \pi_j} D(\mu_j, x_i)^2 \tag{3}$$

The *k*-means algorithm is an iterative algorithm which in every step attempts to further minimize the distortion. Given a set of cluster centroids, the algorithm assigns instances to the cluster with the **nearest** centroid which of course minimizes the distortion. This is Step 1 of the algorithm. We derive Step 2 which recalculates the cluster centroids so as to minimize the distortion in the uni-variate (one dimensional) setting for clarity. This is achieved by taking the first order derivative of the error (Equation 3) with respect to the j^{th} centroid and setting it to zero and solving. A solution to the resulting equation gives us the *k*-means centroid update rule as shown in Equation 5. The multi-variate solution to equation will contain partial derivatives which can be eliminated if modeling assumptions such as column independence from each other area are made.

$$\frac{d(VQE_j)}{d(\mu_j)} = \sum_{x_i \in \pi_j} D(\mu_j, x_i)^2 = 0 \tag{4}$$

$$\mu_j = \sum_{x_i \in \pi_j} x_i / |\pi_j| \tag{5}$$

Recall that π_j is the set of instances closest to the centroid of the j^{th} cluster. These two steps are used in the standard *k*-means algorithm shown in Figure 16.

5.1 COP-*k*-Means

The COP-*k*-means algorithm shown in Figure 17 can be seen to be two part variation of the *k*-means algorithm that incorporates conjunctions of constraints. Firstly, the transitive closure over the must-linked instances is computed with the resultant connected components being replaced by a super-instance whose co-ordinates are the average of the connected component's and whose weight is equal to the number of instances within it (lines 1 and 2). Secondly, rather than performing a nearest centroid assignment (step 2a.1) in Figure 16), a nearest **feasible** centroid assignment is performed (lines 4a.1). When performing the nearest feasible centroid assignment step it is important to remember that the previous set partition is *forgotten* and the new partition built up **incrementally**. Therefore, the first instance assigned to a cluster can **never** violate any constraints, even if it is involved in many. Similarly if there is only one constraint, $c_{\neq}(x,y)$, if x is assigned first then y is assigned to its closest feasible centroid and the assignment of x is not revisted. In this way,

```
Input: $X$: A set of data instances to cluster, $C_=$: set of pairwise must-link constraints, $C_{\neq}$: set of pairwise cannot-link constraints, $k$: the number of clusters to find. Initially, the weight of each instance is 1.
Output: A partition of $X, \Pi_k = \{\pi_1 \cup \pi_k\}$ into $k$ clusters that is a local optima of the VQE (equation 2). and all constraints in $C = C_= \cup C_{\neq}$ are satisfied.

(1) Compute the transitive closure of the set $C_=$ to obtain the connected components $CC_1, \ldots, CC_r$.
(2) For each $i$, $1 \leq i \leq r$, replace all the data instances in $CC_i$ by a single instance with weight $|CC_i|$; the instance's coordinates are obtained by averaging the coordinates of the instances in $CC_i$.
(3) Randomly generate cluster centroids $\mu_1, \ldots, \mu_k$.
(4) loop until convergence do
    . (a) for $i = 1$ to $|X|$ do
    .     (a.1) Assign $x_i$ to the nearest (in terms of distance to centroids) feasible cluster.
    .     (a.2) If assignment of $x_i$ to any cluster always violates a constraint, then exit with failure.
    . (b) Recalculate centroids $\mu_1, \ldots, \mu_k$ taking into account the weight of the instances in $X$ using equation 5
(5) Return with success.
```

Fig. 17. Clustering under Constraints Using COP-k-means

we can view this algorithm as greedily trying to attempt constructing a feasible clustering with *no* backtracking of previous instance assignments.

Natural variations of trying to satisfy all constraints are: a) attempting to satisfy as many constraints as possible while ignoring noisy or inappropriate constraints and b) having degrees of belief/importance associated with each constraint. Both can be viewed as frameworks that allow trading of satisfying lesser important constraints. We now discuss the distance based formulation of this framework and later a probabilistic framework in section 9. Both algorithms have been shown to be useful at ignoring noisy constraints but if all constraints are useful they should perform similarly to COP-k-means.

5.2 Algorithms With Distance Penalties

The COP-k-means algorithm (see section 5.1) can improve the accuracy at predicting an extrinsic label and also shape clusters into desirable forms. However, when constraints are generated from labeled data there is the possibility of class label noise and hence generating a cannot-link or must-link constraint between two instances that should not be. Similarly, if constraints are generated by domain experts, some constraints may be ill-specified or even contradictory. The algorithms in this subsection attempt to ignore noisy or inappropriate constraints by allowing constraints to be left unsatisfied but with a penalty. This involves a trade-off between finding the best clustering and satisfying as many constraints as possible. To achieved this, the penalty of ignoring a constraint must be in the same units as the measure for how good the clustering of the data is. The CVQE (constrained vector quantization error) algorithm discussed in this section uses distance as the fundamental unit and the PKM (Pairwise-constrained k-Means) algorithm discussed later uses probability.

5.2.1 *CVQE.* The core idea behind the CVQE algorithm is to penalize constraint violations using distance. If a must-link constraint is violated then the penalty is the distance between the two centroids of the clusters containing the two instances that should be together. If a cannot-link constraint is violated then the penalty is the distance between the cluster centroid the two instances are assigned to and distance to the nearest cluster centroid. These two penalty types give rise to a new objective function which is termed the

Constrained Vector Quantization Error (CVQE) shown in equation 6.

$$\begin{aligned} CVQE_j \; = \; & \frac{1}{2} \sum_{x_i \in \pi_j} D(\mu_j, x_i)^2 \; + \\ & \frac{1}{2} \sum_{x_i \in \pi_j, (x_i, x_a) \in C_=, g(x_i) \neq g(x_a)} D(\mu_j, \mu_{g(x_a)})^2 \\ & \frac{1}{2} \sum_{x_i \in \pi_j, (x_i, x_a) \in C_{\neq}, g(x_i) = g(x_a)} D(\mu_j, \mu_{h(g(x_a))})^2 \end{aligned} \tag{6}$$

These penalties were found by experimentation to be useful and others [Pelleg and Baras 2007] (see next section) have improved upon these.

The first step of the constrained k-means algorithm must minimize the new constrained vector quantization error. This is achieved by assigning instances so as to minimize the new error term. For instances that are not part of constraints, this involves as before in regular k-means, performing a nearest cluster centroid calculation. For pairs of instances in a constraint, for each possible combination of cluster assignments, the *CVQE* is calculated and the instances are assigned to the clusters that minimally increases the *CVQE*. This new assignment step is shown in equation 7 and requires at most $O(k^2)$ calculations per assignment.

CVQE: Instance Assignment Rule

$$\begin{aligned} \forall x_i \notin C_= \cup C_{\neq} : \;\; & argmin_j D(x_i, \mu_j)^2 \\ \forall (x_a, x_b) \in C_= : & argmin_{i,j} D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \neg\delta(a,b) * D(\mu_j, \mu_i)^2 \\ \forall (x_a, x_b) \in C_{\neq} : & argmin_{i,j} D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \delta(a,b) * D(\mu_j, \mu_{h(\mu_j)})^2 \end{aligned} \tag{7}$$

The second step is to update the cluster centroids so as to minimize the constrained vector quantization error. To achieve this we take the first order derivative of the error, set to zero, and solve. Solving for μ_j, we get the update rule shown in equation 8.

CVQE: Centroid Update Rule

$$\mu_j \; = \; \frac{\sum_{x_i \in \pi_j} [x_i + \sum_{(x_i, x_a) \in C_=, g(x_i) \neq g(x_a)} \mu_{g(x_a)} + \sum_{(x_i, x_a) \in C_{\neq}, g(x_i) = g(x_a)} \mu_{h(g(x_a))}]}{|\mu_j| + \sum_{x_i \in \mu_j, (x_i, x_a) \in C_=, g(x_i) \neq g(x_a)} 1 + \sum_{s_i \in \pi_j, (x_i, x_a) \in C_{\neq}, g(x_i) \neq g(x_a)} 1} \tag{8}$$

The intuitive interpretation of the centroid update rule is that if a must-link constraint is violated, the cluster centroid is moved towards the other cluster containing the other instance. Similarly, the interpretation of the update rule for a cannot-link constraint violation is that cluster centroid containing both constrained instances should be moved to the nearest cluster centroid so that one of the instances eventually gets assigned to it, thereby satisfying the constraint.

5.3 LCVQE: An Extension to CVQE

Pelleg and Baras [Pelleg and Baras 2007] create a variation of the assignment and update rules for CVQE that they term LCVQE. Though there algorithm was not derived to minimize a particular objective function, it showed improved performance over LCVQE on several standard data sets both in terms of accuracy and run-time. The two main extensions made by this algorithm over CVQE are: a) not computing all possible k^2 assignments but only a subset of reasonable assignments and b) Changing the penalty for a cannot-link constraint to be the distance from the most outlying (with respect to the cluster centroid) instance in the CL constraint to the cluster centroid nearest it.

The assignment step shown in equation 9 and the centroid update rule is shown in equation 10.

LVCQE: Instance Assignment Rule

$$\begin{aligned}
\forall x_i \notin C_{=} \cup C_{\neq} :\quad & argmin_j D(x_i, \mu_j)^2 \\
\forall (x_a, x_b) \in C_{=} : argmin_{[i=g(x_a), j=g(x_b)],[i=j=g(x_a)],[i=j=g(x_b)]} & \\
D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \neg\delta(a,b) * D(\mu_j, \mu_i)^2 & \\
\forall (x_a, x_b) \in C_{\neq} : argmin_{[i=g(x_a), j=g(x_b)],[D(x_a, g(x_a)) < D(x_b, g(x_b)) : i=j=g(x_a)]} & \\
D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \delta(a,b) * D(\mu_j, \mu_{g(x_b)})^2 &
\end{aligned} \tag{9}$$

This modified assignment rule has the same must-link penalty as CVQE except that not all k possible cluster assignments are checked. The three combinations checked are to place instances x_i and x_j: a) in their closest clusters respectively, b) together in the cluster closest to x_i and c) together in the cluster closest to x_j. The cannot-link penalty is the distance between the cluster centroid the instances are assigned to (C_*) and the centroid nearest to the most outlying of the pair of points with regard to the distance to C_*.

LVCQE: Centroid Update Rule

$$\mu_j = \frac{\sum_{x_i \in \pi_j} [x_i + \sum_{(x_i,x_a) \in C_{=}, g(x_i) \neq g(x_a)} \mu_{g(x_a)} + \sum_{(x_i,x_a) \in C_{\neq}, g(x_i) = g(x_a), D(x_i) < D(x_a)} \mu_{g(x_a)}]}{|\mu_j| + \sum_{s_i \in \mu_j, (s_i,s_x) \in C_{=}, g(s_i) \neq g(s_x)} 1 + \sum_{s_i \in \mu_j, (s_i,s_x) \in C_{\neq}, g(s_i) \neq g(s_x)} 1} \tag{10}$$

5.4 PKM

The PKM algorithm allows constraints to be violated during clustering, as does CVQE but enforces a probabilistic penalty of constraint violation via use of a prior. The algorithm uses a hidden Markov random field (HMRF) approach to construct a prior such that those clusterings (set partitions) with fewers constraint violations are more probable apriori than those with many. PKM is a special case of the HMRF-KMeans algorithm, which is described in detail in the Section 9 — PKM is an ablation of HMRF-KMeans, doing constraint enforcement but not performing distance learning.

6. INITIALIZING AND GENERATING CONSTRAINTS

6.1 Initialization with constraints

Good initial centroids are essential for the success of partitional clustering algorithms such as KMeans or EM [Basu et al. 2004]. Basu and collaborators use a two stage initialization process (Neighborhood inference and Cluster selection) to get good centroids from both the constraints and the unlabeled data.

Neighborhood inference: The transitive closure of the must-link constraints is taken to get connected components consisting of instances connected by must-links. Let there be λ connected components, which are used to create λ neighborhoods.

Cluster selection: The λ neighborhood sets produced in the first stage are used to initialize the algorithm. If $\lambda = k$, the desired number of clusters, λ cluster centers are initialized with the centroids of all the λ neighborhood sets. If $\lambda < k$, λ clusters are initialized from the neighborhoods, and the remaining $k - \lambda$ clusters are initialized with instances obtained by random perturbations of the overall global centroid, following the methodology of Dhillon et al. [Dhillon et al. 2001]. If $\lambda > k$, a weighted variant of farthest-first traversal [Hochbaum and Shmoys 1985] is applied to the centroids of the λ neighborhoods, where the weight of each centroid is proportional to the size of the corresponding neighborhood. Weighted farthest-first traversal selects neighborhoods that are relatively far apart as well as large in size, and the chosen neighborhoods are set as the k initial cluster centroids.

Overall, this two-stage initialization procedure is able to take into account both unlabeled data and constraints to obtain cluster representatives that provide a good initial partitioning of the data set. The authors show that this initializing procedure with constraints and then running regular unconstrained k-means produces as good cluster accuracies (at predicting an extrinsic label) than COP-k-means.

6.2 Active acquisition of constraints

In some domains the ability to choose which pairs of instances to generate constraints on is available. That is, two instances can be nominated and an Oracle is asked what constraint exists between the two. This situation is particularly important when constraints are obtained by querying a user or domain expert since getting constraints on pairs of data instances can be an expensive process. In order to get pairwise constraints that are more informative than randomly chosen constraints, [Basu et al. 2004] proposed a 2-phase active learning scheme for selecting pairwise constraints by asking queries in an interactive user-driven framework. The goal is to ask the minimal number of queries to get constraints, which, when used to cluster the data will give a better constrained clustering of the data than that obtained using randomly chosen constraints. Getting good initial centroids is critical for the success of greedy algorithms such as KMeans – so the motivation here is to get as many instances as possible for each cluster (proportional to the actual cluster size) by asking pairwise queries, so that the algorithm is initialized from a very good set of centroids. The proposed active learning scheme of Basu and collaborators has two phases: EXPLORE and CONSOLIDATE. We can view the first phase as finding the appropriate underlying skeleton of the underlying clusters by finding a point in each of the k clusters. The consolidate phase then tries to collect more instances for each cluster so that the centroid estimate for each cluster better matches the true centroid of the cluster.

The EXPLORE phase explores the given data using farthest-first traversal [Hochbaum and Shmoys 1985] to get k pairwise disjoint non-null neighborhoods as fast as possible, with each neighborhood belonging to a different cluster in the underlying clustering of the data. Note that even if there is only one instance per neighborhood, this neighborhood structure defines a correct skeleton of the underlying clustering. EXPLORE continues till the algorithm runs out of queries or k pairwise disjoint neighborhoods have been found. In the latter case, active learning enters the consolidation phase.

If end of the EXPLORE phase is reached without running of out queries, then at least one instance has been obtained per cluster. The cluster skeleton obtained from EXPLORE is used to initialize k pairwise disjoint non-null neighborhoods $\{N_p\}_{p=1}^{k}$. Then, given any instance x not in any of the existing neighborhoods, at most $(k-1)$ queries are asked by pairing x up with a member from each of the disjoint neighborhoods N_p to find out the neighborhood to which x belongs. This principle forms the second phase of the active learning algorithm which is called the CONSOLIDATE phase. In this phase, the correct cluster label of x are obtained by asking at most $(k-1)$ queries. The queries will be formed by taking a instance y from each of the neighborhoods in turn and asking for the label on the pair (x,y) until a must-link is obtained. Either a must-link reply is obtained in $(k-1)$ queries, or a cannot-link replies for the $(k-1)$ queries to the $(k-1)$ neighborhoods. It is then inferred that the instance is must-linked to the remaining neighborhood. The details of EXPLORE and CONSOLIDATE are given in Figures 18 and 19.

```
Algorithm: EXPLORE
Input: Set of data instances $X = \{x_i\}_{i=1}^{n}$, access to an oracle that answers pairwise
  queries, number of clusters $k$, total number of queries $Q$.
Output: $\lambda \leq k$ disjoint neighborhoods $N = \{N_p\}_{p=1}^{\lambda}$ corresponding to the true
  clustering of $X$ with at least one instance per neighborhood.
Method:
1. Initialize: set all neighborhoods $N_p$ to null
2. Pick the first instance $x$ at random, add to $N_1$, $\lambda \leftarrow 1$
3. While queries are allowed and $\lambda < k$
      $x \leftarrow$instance farthest from the instances in the existing neighborhoods $N$
      if, while pairing $x$ with a instance from each existing neighborhood and querying,
      it is found that $x$ is cannot-linked to all existing neighborhoods
         $\lambda \leftarrow \lambda + 1$, start a new neighborhood $N_\lambda$ with $x$
      else
         add $x$ to the neighborhood with which it is must-linked
```

Fig. 18. Explore algorithm

Active learning for constrained clustering has not been studied as extensively as active learning for classification, which is a long-studied problem where different principles of query selection have been studied, e.g., reduction of the version space size [Freund et al. 1997], reduction of uncertainty in predicted label [Lewis and Gale 1994], maximizing the margin on training data [Abe and Mamitsuka 1998], finding high variance data instances by density-weighted pool-based sampling [McCallum and Nigam 1998], etc. In the clustering setting, [Hofmann and Buhmann 1998] consider another model of active learning – they have incomplete pairwise similarities between instances, and their active learning goal is to select new data, using expected value of information estimated from the existing data,

```
Algorithm: CONSOLIDATE
Input: Set of data instances $X = \{x_i\}_{i=1}^n$, access to an oracle that answers pairwise
   queries, number of clusters $k$, total number of queries $Q$, $k$ disjoint neighborhoods
   corresponding to true clustering of $X$ with at least one instance per neighborhood.
Output: $k$ disjoint neighborhoods corresponding to the true clustering of $X$ with
   higher number of instances per neighborhood.
Method:
1. Estimate centroids $\{\mu_h\}_{h=1}^k$ of each of the neighborhoods
2. While queries are allowed
2a.   randomly pick a instance $x$ not in the existing neighborhoods
2b.   sort the indices $h$ with increasing distances $\|x - \mu_h\|^2$
2c.   for  $h = 1$ to $k$
          query $x$ with each of the neighborhoods in sorted order till a must-link is
          obtained, add $x$ to that neighborhood
```

Fig. 19. Consolidate algorithm

such that the risk of making wrong estimates about the true underlying clustering from the existing incomplete data is minimized. [Klein et al. 2002] also consider active learning in constrained clustering, but instead of making instance-level queries they make cluster level queries, i.e., they ask the user whether or not two whole clusters should be merged.

7. AGGLOMERATIVE HIERARCHICAL CLUSTERING ALGORITHMS

Hierarchical clustering algorithms are used extensively in many areas of science that wish to capture the natural hierarchical structure in data such as the generation of phylogenetic (evolutionary) trees. These algorithms differ from the algorithms in section 5 by allowing the user to choose a particular clustering granularity. Furthermore, in many domains clusters that naturally occur within other clusters are common. Consider a hierarchy of text documents. The top level would of course be articles, the next divides the articles into sports, business, politics, the next divides the sports cluster into baseball, basketball, hockey etc.

Hierarchical clustering algorithms are typically deterministic and create a **dendrogram**, a tree structure containing a k-block set partition for each value of k between 1 and $|X|$. The popular agglomerative algorithms are easy to implement as they begin with each instance in its own cluster and progressively join the closest clusters to reduce the number of clusters by 1 until $k = 1$. The basic agglomerative hierarchical clustering algorithm is shown in Figure 20. Hierarchical clustering algorithms use a variety of distance measures between clusters. Typically measures include: a) Centroid: the distance between cluster centroids, b) Single-Linkage: the distance between the closest instances in the two clusters and c) Complete-Linkage: the distance between the furthest instances in the two clusters. However, compared to non-hierarchical algorithms that typically have $O(n)$ time complexity, typical implementation of agglomerative hierarchical algorithms uses $O(n^2)$ time, though due to their deterministic nature agglomerative algorithms are typically only run once.

In their paper Davidson and Ravi perform a complexity analysis of the feasibility problem for the hierarchical case. This problem is *significantly* different from the feasibility problems considered in previous work [Davidson and Ravi 2005a] since the value of k for

```
Input: Set $X = \{x_1, x_2, \ldots, x_n\}$ of instances, A distance function, $D(\pi_i, \pi_j)$ between two groups of instances.

Output: Dendrogram$_k$, for each $k$, $1 \leq k \leq n = |X|$.

(1) $\pi_i = \{x_i\}$, $1 \leq i \leq n$. Dendrogram$_n$ = $\{\pi_1, \pi_2, \ldots, \pi_n\}$.
(2) for $k = n-1$ down to 1 do
      . (a) Let $(a,b)$ = argmin$_{(i,j)}\{D(\pi_i, \pi_j) : 1 \leq i < j \leq k+1\}$.
      . (b) Obtain Dendrogram$_k$ from Dendrogram$_{k+1}$ by merging $\pi_b$ into $\pi_a$ and then
      deleting $\pi_b$.
    end for
```

Fig. 20. Standard Agglomerative Clustering Algorithm

hierarchical clustering is not given. Formally:

Definition 2 ***Feasibility problem for Hierarchical Clustering*** *(*FHC*)* Instance: *A set X of nodes, the (symmetric) distance $D(x,y) \geq 0$ for each pair of nodes x_i and x_j in X and a collection C of constraints.*

Question: *Can X be partitioned into subsets (clusters) so that all the constraints in C are satisfied?*

The complexity results (Table V) for this problem are quite different than the non-hierarchical case since it is more relaxed in that bounds on k are not given. However, we see that another issue is raised: namely that of dead-ends. Dead-ends can be explained by the following case: If we are given a feasible clustering with k_{max} clusters and even if we know there is another clustering with k_{min} clusters then algorithms that simply join the two closest clusters may yield a feasible but "dead-end" solution with k clusters where $k_{max} < k < k_{min}$. Figure 21 gives an example of a situation leading to a dead-end for six points. The closest cluster join strategy will join E with D and then DE with F but there are no further joins possible due to constraints causing the algorithm to terminate for $k = 4$. But there exists other clusterings with $k = 3$ such as $\{C, BE, ADF\}$. Therefore, traditional nearest joins algorithms may create dendrograms that are incomplete. How to overcome dead-ends is an important open question.

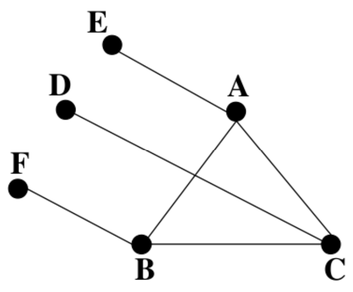


Fig. 21. Example of CL constraints (lines) that lead to dead-end. Note the instance positions in figure reflect distances between points, for example points D and E are the closest.

Constraint	Unspecified k	Unspecified k - Dead-ends?
Must-Link	**P**	No
Cannot-Link	**P**	Yes
δ-constraint	**P**	No
ϵ-constraint	**P**	No
Must-Link and δ	**P**	No
Must-Link and ϵ	**P**	No
δ and ϵ	**P**	No
Must-Link, Cannot-Link, δ and ϵ	**NP**-complete	Yes

Table V. Results for Feasibility Problems for Unspecified k (hierarchical clustering). Compare with Tables I and II.

Davidson and Ravi explore changing agglomerative hierarchical clustering algorithms to satisfy **all** instance-level and cluster-level constraints as discussed in the next sub-section. However, hierarchical clustering has a long history of using spatial constraints to find specific types of clusters and avoid others [Yang et al. 2001; Zaiane et al. 2000] the instance level constraints we describe can specify a variety of spatial constraints (i.e. Figure 1) but also constraints on pairs of instances, features etc. Instance-level constraints have also been used to learn a distance *matrix* for agglomerative clustering [Klein et al. 2002] (discussed in a later sub-section); but the agglomerative algorithms are themselves not altered to satisfy the constraints.

7.1 Agglomerative Algorithms To Satisfy All ML and CL Constraints

The change to the basic agglomerative algorithm (Figure 20) to satisfy all ML and CL constraints is shown in Figure 22. The algorithm starts off by computing the transitive closure and rather than starting with n clusters (one for each instance) there are $|X1|+r$ clusters where $|X1|$ is the number of instances not involved in a ML constraint and r the number of connected components. Joins are then performed so that the closest two clusters whose join does not result in a constraint violation are merged. Since the feasibility results in Table V show that when using most combinations of constraints this algorithm will always find a feasible solution, the same table shows that such an algorithm may not always generate a full dendrogram. That is, there may be another series of joins (beyond the closest feasible combination) that will give a more complete/full dendrogram.

7.1.1 *Using a γ Constraint To Improve Run-time Using the Triangle Inequality.* Davidson and Ravi also introduce a new constraint, the γ constraint and illustrate how the triangle inequality can be used to further improve the run-time performance of agglomerative hierarchical clustering using a centroid distance function. They show though no worst-case improvement occurs, empirically the constraint does improve performance and provide a probabilistic expected performance improvement using the Markov inequality. There exists other work involving the triangle inequality for non-hierarchical clustering [Elkan 2003] as well as for hierarchical clustering [Nanni 2005] but neither make use of constraints.

The γ constraint allows specifying how geometrically well separated the clusters should be and is related to the δ constraint in Figure 1.

```
ConstrainedAgglomerative($X$,$C_=$,$C_{\neq}$) returns $Dendrogram_i$, $i = k_{min}$ ... $k_{max}$ such that each level in the dendrogram satisfies all constraints.

Notes: In Step 5 below, the term "mergeable clusters" is used to denote a pair of clusters whose merger does not violate any of the given CL constraints. The value of $t$ at the end of the loop in Step 5 gives the value of $k_{\min}$.

(1) Construct the transitive closure of the constraints in $C_=$ (see [Davidson and Ravi 2005a] for an algorithm)
    resulting in $r$ connected components $M_1, M_2, \ldots, M_r$.
(2) If two instances $\{x_i, x_j\}$ are both a CL and ML constraint then output "No Solution" and stop.
(3) Let $\mathbf{X}_1 = X - (\bigcup_{i=1}^{r} M_i)$. Let $k_{\max} = r + |\mathbf{X}_1|$.
(4) Construct an initial feasible clustering with $k_{\max}$ clusters consisting of the $r$ clusters $M_1, \ldots, M_r$ and a
    singleton cluster for each instance in $\mathbf{X}_1$. Set $t = k_{\max}$.
(5) while (there exists a pair of mergeable clusters) do
    . (a) Select a pair of clusters $\pi_l$ and $\pi_m$ according to the specified distance criterion.
    . (b) Merge $\pi_l$ into $\pi_m$ and remove $\pi_l$. (The result is $Dendrogram_{t-1}$.)
    . (c) $t = t - 1$.
    endwhile
```

Fig. 22. Agglomerative Clustering with ML and CL Constraints

```
IntelligentDistance ($\gamma$, $\Pi = \{\pi_1, \ldots, \pi_k\}$)
returns $D(x_i, x_j)$ $\forall i, j$.
(1) for $i = 2$ to $n-1$   $D_{x_1,x_i} = D(\pi_1, \pi_i)$ endloop
(2) for $i = 2$ to $n-1$
      for $j = i+1$ to $n-1$   $\hat{D}_{x_i,x_j} = |D_{x_1,x_i} - D_{x_1,x_j}|$
       if $\hat{D}_{x_i,x_j} > \gamma$ then $D_{x_i,x_j} = \gamma + 1$ ; do not join   else $D_{x_i,x_j} = D(x_i, x_j)$
      endloop
    endloop
(3) return $D_{x_i,x_j}$, $\forall i, j$.
```

Fig. 23. Function for Calculating Distances Using the γ Constraint and the Triangle Inequality.

Definition 3 *(The γ Constraint For Hierarchical Clustering) Two clusters whose geometric centroids are separated by a distance greater than γ cannot be joined.*

Recall that the triangle inequality for three instances a, b, c refers to the expression $|D(a,b) - D(b,c)| \leq D(a,c) \leq D(a,b) + D(c,b)$ where D is the Euclidean distance function or any other metric function. We can improve the efficiency of the hierarchical clustering algorithm by making use of the lower bound in the triangle inequality and the γ constraint. Let a, b, c now be cluster centroids and we wish to determine the closest two centroids to join. If we have already computed $D(a,b)$ and $D(a,c)$ and the value $|D(a,b) - D(a,c)|$ *exceeds* γ, then we need not compute the distance between b and c as the lower bound on $D(b,c)$ already exceeds γ and hence b and c cannot be joined. Formally the function to calculate distances using geometric reasoning at a particular dendrogram level is shown in Figure 23. Central to the approach is that the distance between a central instance (a) (in this case the first) and every other instance is calculated. Therefore, when bounding the distance between two instances (b, c) we effectively calculate a triangle with two edges with know lengths incident on a and thereby lower bound the distance between b and c. How to select the best central instance and the use of multiple central instances remains future important research though some progress is being made [Nanni 2005].

If the triangle inequality bound exceeds γ, then we save making d floating-point power calculations since the data instances are in d dimensional space. We have no reason to believe that there will be at least one situation where the triangle inequality saves computation in *all problem instances*; hence in the worst case, there is no performance improvement. But in practice it is expected to occur and Davidson and Ravi derive an expected case improvement bound using the Markov inequality.

Observation 8 ***Expected Case Improvement Using γ Constraint***. *The γ constraint states that two clusters whose centroids are more than γ distance apart should not be joined together. For an data set of size n, the number of distance calculations to complete a full dendrogram will be $O(n^2)$. If $\gamma = c\rho$ where ρ is the average distance between any two instances in the data set then the expected number of calculations will be $O(n^2 - \frac{n^2}{2c})$.*

To illustrate this in example, consider the 150 instance IRIS data set (n=150) where the average distance (with attribute value ranges all being normalized to between 0 and 1) between two instances is 0.6; that is, $\rho = 0.6$. If we state that we do not wish to join clusters whose centroids are separated by a distance greater than 3.0, then $\gamma = 3.0 = 5\rho$. By not using the γ constraint the total number of computations is 22201. But by using the γ constraint on average (over all data sets whose average pairwise point distance is 0.6) the number of computations that are saved is at least $\frac{1}{2c}$ hence the saving is about 10%. Davidson and Ravi then go onto show that the γ constraint can be used to improve efficiency of the basic agglomerative clustering algorithm for many data sets and the actual performance improvement typically exceeds the bound given the Markov inequality is a weak bound.

7.2 Changing the Distance Matrix

The work on Klein and collaborators [Klein et al. 2002] explores the problem of changing the distance matrix to reflect the constraints. In this work $c_=(a,b)(c_{\neq}(a,b))$ indicates that only instances a and b should be made closer together (further apart). Later in section 8) we shall discuss work that learns a distance *metric* so as to change the distance between a and b and instances surrounding these two instances.

Learning a distance matrix is a multi-step pre-processing algorithm that produces a distance matrix (D') that can be used with any hierarchical clustering algorithm. The algorithm is shown in Figure 24. The first two steps create a new distance matrix using the Euclidean distance between the points and in step three make all must-linked instances have a distance of 0. However, this change ruins the triangle inequality and hence the resultant distance matrix does not represent a metric. Step four corrects this by performing the shortest path calculations but this step takes $O(n^3)$ time. Finally, the last step (which can invalidate the triangle inequality) is to make the cannot-linked instances far apart. Klein and collaborators argue that due to the entailment property of CL constraints (see 2) that the triangle inequality is effectively maintained. Consider a four instance data set (x_1, x_2, x_3 and x_4), two constraints $c_=(x_1,x_2)$ and $c_{\neq}(x_1,x_4)$ and the distance matrix:

$$D = \begin{pmatrix} 0 & 3 & 6 & 1 \\ 3 & 0 & 3 & 2 \\ 6 & 3 & 0 & 5 \\ 1 & 2 & 5 & 0 \end{pmatrix}$$

After step three the modified distance matrix looks like:

```
*CreateDistanceMatrix*
input: A set of must-link $C_=$ and cannot-link $C_{\neq}$ constraints, a set of data instances $X$
output A modified distance matrix $D'(x_i,x_j)\ \forall i,j$.
(1) Calculate the Euclidean distance between all instances and store in matrix $D_{i,j} = D_{j,i} = D(x_i,x_j)\forall i,j$
(2) Initialized modified distance matrix $D'$ with $D$
(3) $\forall c_=(x_i,x_j) \in C_= : D'_{i,j} = D'_{j,i} = 0$
(4) $\forall x_i,x_j\ D'_{i,j} = D'_{j,i} = ShortestPath(i,j)$ using $D'$
(5) $\forall c_{\neq}(x_i,x_j) \in C_{\neq} : D'_{i,j} = D'_{j,i} = max(D)+1$
```

Fig. 24. Algorithm for Creating a New Distance Matrix for Agglomerative Algorithms.

$$D' = \begin{pmatrix} 0 & \mathbf{0} & 6 & 1 \\ \mathbf{0} & 0 & 3 & 2 \\ 6 & 3 & 0 & 5 \\ 1 & 2 & 5 & 0 \end{pmatrix}$$

After step four the modified distance matrix looks like:

$$D' = \begin{pmatrix} 0 & 0 & \mathbf{3} & 1 \\ 0 & 0 & 3 & \mathbf{1} \\ \mathbf{3} & 3 & 0 & 5 \\ 1 & \mathbf{1} & 5 & 0 \end{pmatrix}$$

After step five the modified distance matrix looks like:

$$D' = \begin{pmatrix} 0 & 0 & 3 & \mathbf{6} \\ 0 & 0 & 3 & 1 \\ 3 & 3 & 0 & 5 \\ \mathbf{6} & 1 & 5 & 0 \end{pmatrix}$$

A valid criticism of this approach is that step five ruins the triangle inequality that step four fixes up. However, the authors argue that the inequality is effectively enforced as the following example indicates. Clearly the distance $D(x_1,x_4)$ violates the triangle inequality since it is larger than $\mid D(x_1,x_3)+D(x_3,x_4) \mid$ indicating that either $D(x_1,x_3)$ or $D(x_3,x_4)$ is too large. Now consider at the first iteration the hierarchical algorithm will join x_1 and x_2, but it will not join this new cluster to x_4 since it is too far away but instead x_3 and hence indirectly we have made $D(x_1,x_3)$ smaller.

8. LEARNING DISTANCE FUNCTION ALGORITHMS

In this section, we will discuss two popular approaches of using constraints for distance metric learning in constrained clustering. While both forms cast the problem of learning a distance metric from constraints as an optimization problem, the former uses a linear algebra formulation while the later a probabilistic formulation.

8.1 Generalized Mahanabolis Distance Learning

[Xing et al. 2003] proposed a formulation for learning a parametrized Mahanabolis metric from ML and CL constraints by exploiting the form $d(x_1,x_2) = \sqrt{(x_1-x_2)^T A(x_1-x_2)}$ where the matrix A represents the distance matrix. A matrix $A = I$ where I is the identify matrix leads to the Euclidean matrix. The authors proposed the following semi-definite program (SDP) for the problem:

$$\begin{aligned} \min_A \sum_{(x_i,x_j)\in ML} ||x_i-x_j||_A^2 &= \min_A \sum_{(x_i,x_j)\in ML} (x_i-x_j)^T A(x_i-x_j) \quad (11) \\ s.t., \sum_{(x_i,x_j)\in CL} ||x_i-x_j||_A &\geq 1, A \succeq 0 \\ where||x_i-x_j||_A = (x_i-x_j)^T A(x_i-x_j) \end{aligned}$$

Minimizing Equation 11 produces a matrix A (which represents the distance metric as a transformation on the original space) such that the must-link instances are brought closer together, while ensuring that the cannot-link instances are kept apart and the underlying metric still satisfies the triangle inequality by ensuring that A is positive semi definite. Equation 11 can be minimized by minimizing the alternate formulation shown in equation 12 since the logarithm of second term will be minimized when it is greater than 1 as required in equation 11.

$$\begin{gathered} \min_A \sum_{(x_i,x_j)\in ML} ||x_i-x_j||_A^2 - log \sum_{(x_i,x_j)\in CL} ||x_i-x_j||_A \quad (12) \\ where||x_i-x_j||_A = (x_i-x_j)^T A(x_i-x_j) \end{gathered}$$

If A is restricted to being a diagonal matrix this term has the benefit of being easily differentiable and convex, hence, Newton-Raphson can be used to find the global optimum. This is effectively learning a generalized Mahanabolis distance where the entry $a_{i,i}$ stretches ($a_{i,i} > 1$) or compresses ($a_{i,i} < 1$) the i^{th} dimension. If $A = I$ where I is the identify matrix then the learnt distance matrix is the Euclidean distance.

However, Newton-Raphson cannot be used if A is not restricted to being diagonal. Instead [Xing et al. 2003] proposed an equivalent formulation of Equation 11 shown in equation 13.

$$\begin{aligned} \max_A g(A) &= \sum_{(x_i,x_j)\in CL} ||x_i,x_j||_A \quad (13) \\ s.t., f(A) &= \sum_{(x_i,x_j)\in ML} ||x_i,x_j||_A^2 A \leq 1 \rightarrow C_1 \quad (14) \\ A &\succeq 0 \rightarrow C_2 \end{aligned}$$

[Xing et al. 2003] optimized Equation 13 using an alternate maximization algorithm, that had 2 steps: (1) gradient ascent – to optimize the objective; (2) iterated projection algorithm – to satisfy the inequality constraints. [Bie et al. 2003] used a variant of Linear Discriminant Analysis (LDA) to find the a Mahanabolis metric from constraints more efficiently than using an SDP.

8.2 Kernel Distance Functions Using AdaBoost

[Hertz et al. 2004] proposed a method for distance metric learning by using boosting in the product space of the input data space X. They posed the constrained metric learning problem as learning a function that took as input the instances in the product space $X \times X$, and output binary labels corresponding to must-link (1) and cannot-link constraints (0). They used boosting on the product space to learn this function, where boosting is a standard machine learning tool that combines the strength of an ensemble of “weak” learners (with low prediction accuracy) to create a “strong” learner (with high prediction accuracy) [Freund and Schapire 1996]. The overall flow of the *DistBoost* algorithm of [Hertz et al. 2004] is outlined in Figure 25. In the first step, a constrained weighted EM algorithm is run on the data set and constraints, to fit a Gaussian Mixture Model (GMM) over weighted unlabeled data and the given constraints. The key difference of constrained EM from ordinary EM is the E-step, which sums the assignment probabilities only over assignments that comply with the constraints. The output of the GMM is treated as a “weak” learner and is used to learn a “weak” distance function, where the distance $h(x_1,x_2)$ between two instances x_1 and x_2 is computed from their MAP assignment in the GMM as follows:

$$h(x_1,x_2) = \max_i p(y_1 = i|\Theta) \cdot \max_i p(y_2 = i|\Theta)$$

The *DistBoost* algorithm computes the weights of the “weak” distance functions using Boosting, and updates the weights on pairs of instances, which are translated to weights on individual data instances. This is again passed back to the input of the GMM-EM algorithm, and the process is repeated for multiple steps.

9. SATISFYING CONSTRAINTS AND LEARNING DISTANCE FUNCTION ALGORITHMS

As mentioned in Section 2, there have been some algorithms that try to both enforce constraints and learn distance functions from constraints for partitional clustering algorithms. In this section we will outline an example of such algorithm, which uses the framework of a generative probabilistic model, the Hidden Markov Random Field (HMRF) [Basu et al. 2004].

9.1 HMRF Model

The Hidden Markov Random Field (HMRF) is a probabilistic generative model for semi-supervised constrained clustering, consisting of the following components: (1) an *observable* set $X = (x_1,\ldots,x_n)$ of random variables, corresponding to the given data instances X; (2) an *unobservable* (hidden) set $Y = (y_1,\ldots,y_n)$ of random variables, corresponding to cluster assignments of instances in X, $y_i \in (1,\ldots,K)$; (3) an *unobservable* (hidden) set of generative model parameters Θ, which consists of distortion measure parameters A (typically a matrix or vector of weights) and cluster representatives $M = (\mu_1,\ldots,\mu_K)$: $\Theta = \{A,M\}$; (4) an *observable* set of constraint variables $C = (c_{12},c_{13},\ldots,c_{n-1,n})$. Each c_{ij} is a tertiary variable taking on a value from the set $(-1,0,1)$, where $c_{ij} = 1$ indicates that $(x_i,x_j) \in C_{ML}$, $c_{ij} = -1$ indicates that $(x_i,x_j) \in C_{CL}$, and $c_{ij} = 0$ corresponds to pairs (x_i,x_j) that are not constrained. The constraints are accompanied by associated violation costs W, where w_{ij} represents the cost of violating the constraint between instances x_i and x_j if such a constraint exists. Fig. 26 shows a simple example of an HMRF having five

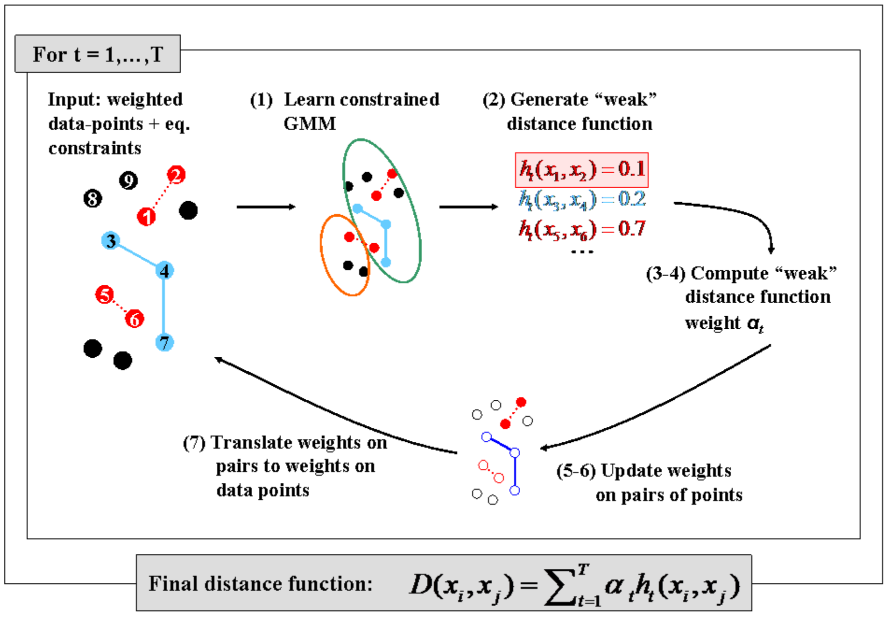

For t = 1,…,T
Input: weighted data-points + eq. constraints
(1) Learn constrained GMM
(2) Generate "weak" distance function
$h_t(x_1,x_2)=0.1$
$h_t(x_3,x_4)=0.2$
$h_t(x_5,x_6)=0.7$
(3-4) Compute "weak" distance function weight α_t
(5-6) Update weights on pairs of points
(7) Translate weights on pairs to weights on data points
Final distance function: $D(x_i,x_j)=\sum_{t=1}^{T}\alpha_t h_t(x_i,x_j)$

Fig. 25. DistBoost algorithm

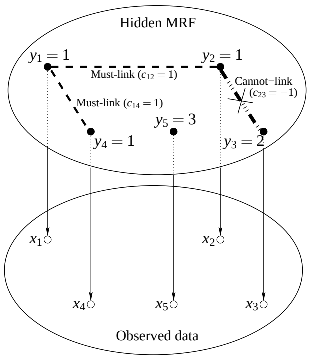


Fig. 26. A Hidden Markov Random Field

data instances partitioned into three clusters, while maximally respecting three pairwise constraints.

The joint probability of X, Y, and Θ, given C, in the described HMRF model can be factorized as follows:

$$\mathrm{P}(X,Y,\Theta|C) = \mathrm{P}(\Theta|C)\ \mathrm{P}(Y|\Theta,C)\ \mathrm{P}(X|Y,\Theta,C) \tag{15}$$

The graphical plate model [Buntine 1994] of the dependence between the random variables in the HMRF is shown in Figure 27. The prior probability of Θ is assumed to be independent of C. The probability of observing the label configuration Y depends on the

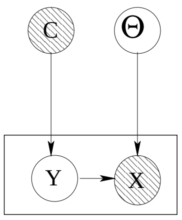


Fig. 27. Graphical plate model of variable dependence

constraints C and current generative model parameters Θ. Observed data instances corresponding to variables X are generated using the model parameters Θ based on cluster labels Y, independent of the constraints C. The variables X are assumed to be mutually independent: each x_i is generated individually from a conditional probability distribution $\mathrm{P}(x_i|y_i,\Theta)$.

[Basu et al. 2004] show that the joint probability on the HMRF is equivalent to maximizing:

$$\mathrm{P}(X,Y,\Theta|C) = \mathrm{P}(\Theta)\left(\frac{1}{Z}\exp\big(-\sum_{c_{ij}\in C} v(i,j)\big)\right)\left(\prod_{i=1}^{n} p(x_i|y_i,\Theta)\right) \tag{16}$$

They chose the following Gibbs potential for $\mathrm{P}(Y|\Theta,C)$:

$$\mathrm{P}(Y|\Theta,C) = \frac{1}{Z}\exp(-\sum_{i,j} v(i,j)), \tag{17}$$

where each constraint potential function $v(i,j)$ has the following form inspired by the generalized Potts model [Kleinberg and Tardos 1999]:

$$v(i,j) = \begin{cases} w_{ij} f_{ML}(i,j) & \text{if } c_{ij}=1 \text{ and } y_i \neq y_j \\ w_{ij} f_{CL}(i,j) & \text{if } c_{ij}=-1 \text{ and } y_i = y_j \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

The joint probability formulation in Eq.(16) provides a general framework for incorporating various similarity functions in clustering by choosing a particular form of $p(x_i|y_i,\Theta)$, the probability density that generates the i-th instance x_i from cluster y_i. [Basu et al. 2004] restrict their attention to probability densities from the exponential family, where the conditional density for observed data can be represented as follows:

$$p(x_i|y_i,\Theta) = \frac{1}{Z_\Theta}\exp\big(-D(x_i,\mu_{y_i})\big), \tag{19}$$

where $D(x_i,\mu_{y_i})$ is the Bregman divergence between x_i and μ_{y_i}, corresponding to the exponential density p, and Z_Θ is the normalizer [Banerjee et al. 2005]. Different clustering models fall into this exponential form:

—If x_i and μ_{y_i} are vectors in Euclidean space, and D is the square of the L_2 distance parametrized by a positive semi-definite weight matrix A, $D(x_i,\mu_{y_i}) = \|x_i-\mu_{y_i}\|_A^2$, then the cluster conditional probability is a d-dimensional multivariate normal density with covariance matrix A^{-1}: $p(x_i|y_i,\Theta) = \frac{1}{(2\pi)^{d/2}|A|^{-1/2}} \exp(-\frac{1}{2}(\|x_i - \mu_{y_i}\|_A^2)$ [Kearns et al. 1997];

—If x_i and μ_{y_i} are probability distributions, and D is KL-divergence ($D(x_i,\mu_{y_i}) = \sum_{m=1}^{d} x_{im} \log \frac{x_{im}}{\mu_{y_i m}}$), then the cluster conditional probability is a multinomial distribution [Dhillon and Guan 2003].

The relation in Eq.(19) holds even if D is not a Bregman divergence but a directional distance measure such as cosine distance. Then, if x_i and μ_{y_i} are vectors of unit length and D is one minus the dot-product of the vectors $\left(D(x_i,\mu_{y_i}) = 1 - \frac{\sum_{m=1}^{d} x_{im}\mu_{y_i m}}{\|x_i\|\|\mu_{y_i}\|}\right)$, then the cluster conditional probability is a von-Mises Fisher (vMF) distribution with unit concentration parameter [Banerjee et al. 2005], which is the spherical analog of a Gaussian.

Putting Eqn. (19) into Eqn. (16) and taking logarithms gives the following cluster objective function, minimizing which is equivalent to maximizing the joint probability over the HMRF in Eqn. (16):

$$\mathcal{J}_{\text{obj}} = \sum_{x_i \in X} D(x_i,\mu_{y_i}) + \sum_{c_{ij} \in C} v(i,j) - \log \mathrm{P}(\Theta) + \log Z + n \log Z_\Theta \tag{20}$$

[Basu et al. 2004] used Rayleigh priors for $\mathrm{P}(\Theta)$, and the ignored the normalizer terms. An optimal clustering is obtained by minimizing $\mathcal{J}_{\text{obj}}$ over the hidden variables Y and parameters Θ, which are comprised of cluster centroids M and distortion measure parameters A (note that given the cluster assignments Y, the means $M = \{\mu_i\}_{i=1}^{K}$ are uniquely determined).

9.2 EM algorithm

As discussed in Section 9.1, [Basu et al. 2004] minimize $\mathcal{J}_{\text{obj}}$ using a K-Means-type iterative algorithm HMRF-KMeans. The outline of the algorithm is presented in Figure 28. The basic idea of HMRF-KMeans is as follows: the constraints are used to obtain a good initialization of the clustering. Then in the E-step, given the current cluster representatives, every data instance is re-assigned to the cluster that minimizes its contribution to $\mathcal{J}_{\text{obj}}$. The E-step of HMRF-KMeans uses an Iterated Conditional Modes (ICM) approach, which is a greedy strategy to sequentially update the cluster assignment of each instance, keeping the assignments for the other instances fixed. In the M-step, the cluster representatives $M = (\mu_1,\ldots,\mu_K)$ are re-estimated from the cluster assignments to minimize $\mathcal{J}_{\text{obj}}$ for the current assignment. The clustering distortion measure, D, is subsequently updated in the M-step to reduce the objective function by modifying the parameters of the distortion measure.

Note that this corresponds to the generalized EM algorithm [Neal and Hinton 1998; Dempster et al. 1977], where the objective function is reduced but not necessarily minimized in the M-step. Effectively, the E-step minimizes $\mathcal{J}_{\text{obj}}$ over cluster assignments Y, the M-step (A) minimizes $\mathcal{J}_{\text{obj}}$ over cluster representatives M, and the M-step (B) reduces $\mathcal{J}_{\text{obj}}$ over the parameters of the distortion measure D. The E-step and the M-step are repeated till a specified convergence criterion is reached. [Basu et al. 2004] show that HMRF-KMeans converges to a local optimum of $\mathcal{J}_{\text{obj}}$.

```
Algorithm: HMRF-KMeans
Input: Set of data points $X = \{x_i\}_{i=1}^n$, number of clusters $K$,
    set of must-link constraints $C_{ML} = \{(x_i, x_j)\}$,
    set of cannot-link constraints $C_{CL} = \{(x_i, x_j)\}$,
    distortion measures $\{D_h\}_{h=1}^K$, constraint violation costs $W$.
Output: Disjoint $K$-partitioning $\{X_h\}_{h=1}^K$ of $X$ such that
    objective function $\mathcal{J}_{\mathrm{obj}}$ in Eqn.(9) is (locally) minimized.
Method:
1. Initialize the $K$ clusters centroids $\{\mu_h^{(0)}\}_{h=1}^K$, set $t \leftarrow 0$
2. Repeat until convergence
2a.   E-step: Given $\{\mu_h^{(t)}\}_{h=1}^K$, re-assign cluster labels
      $\{y_i^{(t+1)}\}_{i=1}^n$ on the points $\{x_i\}_{i=1}^n$ to minimize $\mathcal{J}_{\mathrm{obj}}$.
2b.   M-step(A): Given cluster labels $\{y_i^{(t+1)}\}_{i=1}^n$, re-calculate
      cluster centroids $\{\mu_h^{(t+1)}\}_{h=1}^K$ to minimize $\mathcal{J}_{\mathrm{obj}}$.
2c.   M-step(B): Re-estimate distortion measures $\{D_h\}_{h=1}^K$ to reduce $\mathcal{J}_{\mathrm{obj}}$.
2d.   t ← t+1
```

Fig. 28. HMRF-KMEANS algorithm

9.3 Improvements to HMRF-KMEANS

There have been multiple improvements to the initial HMRF-based probabilistic generative constrained clustering framework. [Lange et al. 2005] incorporated prior knowledge from both labels on the input data instances as well as constraints into their clustering model. They inferred the constraint potentials in the HMRF model from a Maximum Entropy solution of $P(Y)$ under constraints encoded in the label and constraint set, and replaced the ICM-based greedy assignment scheme in the E-step of HMRF-KMEANS by mean-field approximation. [Lu and Leen 2005] proposed probabilistic EM-style assignments instead of winner-take-all KMeans-type assignments, and used Gibbs sampling in the E-step of their constrained EM algorithm.

10. CONCLUSIONS AND OTHER WORK

Clustering with constraints is a rapidly developing area of research. We have covered in detail in this tutorial a sample of the typical approaches used by researchers and contrasted these approaches with other published work. The approaches discussed in detail have the advantage of having accompanying freely available source code. More advanced research in the field will shortly appear as an edited book [Basu et al. 2008].

ACKNOWLEDGMENTS

The first author wishes to thank the NSF for their support of this work in the form of GRANT IIS:0643668 CAREER:Knowledge Enhanced Clustering with Constraints.

We would both like to think Kiri Wagstaff whom we have had had many insightful discussions with. Tomer Hertz, Sepandar Kamvar and Brian Kulis provided us figures that have been reproduced in this paper but all errors remain our own.

11. NOTATION AND SYMBOLS

Sets of Numbers

$\mathbb{N}$	the set of natural numbers, $\mathbb{N} = \{1, 2, \ldots\}$
$\mathbb{R}$	the set of reals
$[n]$	compact notation for $\{1, \ldots, n\}$
$x \in [a,b]$	interval $a \le x \le b$
$x \in (a,b]$	interval $a < x \le b$
$x \in (a,b)$	interval $a < x < b$
$\|C\|$	cardinality of a set C (for finite sets, the number of elements)

Data

$\mathcal{X}$	the input domain
d	(used if $\mathcal{X}$ is a vector space) dimension of $\mathcal{X}$
k^*	number of underlying classes in the labeled data
k	number of clusters (can be different from k^*)
l, u	number of labeled, unlabeled training instances
n	total number of instances, $n = l + u$.
i, j	indices, often running over $[n]$ or $[k]$
x_i	input data instance $x_i \in \mathcal{X}$
y_j	output cluster label $y_j \in [K]$
X	a sample of input data instances, $X = (x_1, \ldots, x_n)$ and $X = \{X_l \cup X_u\}$
Y	output cluster labels, $Y = (y_1, \ldots, y_n)$ and $Y = \{Y_l \cup Y_u\}$
Π_k	k block clustering (set partition) on X: $\{\pi_1, \pi_2 \ldots \pi_k\}$
$\mu_1 \ldots \mu_k$	the k centroids of the k blocks forming the set partition on X
$D(x,y)$	distance between instances x and y
X_l	labeled part of X, $X_l = (x_1, \ldots, x_l)$
Y_l	part of Y where labels are specified, $Y_l = (y_1, \ldots, y_l)$
X_u	unlabeled part of X, $X_u = (x_{l+1}, \ldots, x_{l+u})$
Y_u	part of Y where labels are not specified, $Y_u = (y_{l+1}, \ldots, y_{l+u})$
C	set of constraints
W	weights on constraints
$C_=$	conjunction of must-link constraints
$C_{\neq}$	conjunction of cannot-link constraints
$c_=(i,j)$	must-link constraint between x_i and x_j
$c_{\neq}(i,j)$	cannot-link constraint between x_i and x_j
$w_=(i,j)$	weight on must-link constraint $c_=(i,j)$
$w_{\neq}(i,j)$	weight on cannot-link constraint $c_{\neq}(i,j)$

Kernels

$\mathcal{H}$	feature space induced by a kernel
Φ	feature map, $\Phi : \mathcal{X} \to \mathcal{H}$
K	kernel matrix or Gram matrix, $K_{ij} = k(x_i, x_j)$

Vectors, Matrices and Norms

$\mathbf{1}$	vector with all entries equal to one
$\mathbf{I}$	identity matrix
$A^\top$	transposed matrix (or vector)
A^{-1}	inverse matrix (in some cases, pseudo-inverse)
$\mathbf{tr}\,(A)$	trace of a matrix
$\mathbf{det}\,(A)$	determinant of a matrix
$\langle \mathbf{x}, \mathbf{x}' \rangle$	dot product between $\mathbf{x}$ and $\mathbf{x}'$
$\lVert\cdot\rVert$	2-norm, $\lVert\mathbf{x}\rVert := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
$\lVert\cdot\rVert_p$	p-norm , $\lVert\mathbf{x}\rVert_p := \left(\sum_{i=1}^N \lvert x_i\rvert^p\right)^{1/p}$, $N \in \mathbb{N} \cup \{\infty\}$
$\lVert\cdot\rVert_\infty$	∞-norm , $\lVert\mathbf{x}\rVert_\infty := \sup_{i=1}^N \lvert x_i\rvert$, $N \in \mathbb{N} \cup \{\infty\}$

Functions

$\ln$	logarithm to base e
$\log_2$	logarithm to base 2
f	a function, often from $\mathcal{X}$ or $[n]$ to $\mathbb{R}$, $\mathbb{R}^M$ or $[M]$
$g(x_i)$	a function that returns the closest cluster index to instance $\mathbf{x}_i$
$h(\mu_i)$	a function that returns the closest cluster index to cluster centroid μ_i
$\mathcal{F}$	a family of functions
$L_p(\mathcal{X})$	function spaces, $1 \leq p \leq \infty$

Probability

$\mathrm{P}\{\cdot\}$	probability of a logical formula
$\mathrm{P}(C)$	probability of a set (event) C
$p(x)$	density evaluated at $x \in \mathcal{X}$
$\mathbf{E}\,[\cdot]$	expectation of a random variable
$\mathbf{Var}\,[\cdot]$	variance of a random variable
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2

	Graphs
g	graph g $= (V, E)$ with nodes V and edges E
$\mathcal{G}$	set of graphs
$\mathbf{W}$	weighted adjacency matrix of a graph ($\mathbf{W}_{ij} \neq 0 \Leftrightarrow (i, j) \in E$)
$\mathbf{D}$	(diagonal) degree matrix of a graph, $\mathbf{D}_{ii} = \sum_j W_{ij}$
$\mathcal{L}$	normalized graph Laplacian, $\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
L	un-normalized graph Laplacian, $L = \mathbf{D} - \mathbf{W}$
	Miscellaneous
I_A	characteristic (or indicator) function on a set A i.e., $I_A(x) = 1$ if $x \in A$ and 0 otherwise
δ_{ij}	Kronecker δ ($\delta_{ij} = 1$ if $i = j, 0$ otherwise)
δ_x	Dirac δ, satisfying $\int \delta_x(y) f(y) dy = f(x)$
$O(g(n))$	a function $f(n)$ is said to be $O(g(n))$ if there exist constants $C > 0$ and $n_0 \in \mathbb{N}$ such that $\lvert f(n) \rvert \leq Cg(n)$ for all $n \geq n_0$
$o(g(n))$	a function $f(n)$ is said to be $o(g(n))$ if there exist constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $\lvert f(n) \rvert \geq cg(n)$ for all $n \geq n_0$
rhs/lhs	shorthand for “right/left hand side”
■	the end of a proof

REFERENCES

ABE, N. AND MAMITSUKA, H. 1998. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*. 1–10.

BANERJEE, A., DHILLON, I., GHOSH, J., AND SRA, S. 2005. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research 6*, 1345–1382.

BANERJEE, A., MERUGU, S., DHILON, I., AND GHOSH, J. 2005. Clustering with Bregman divergences. *Journal of Machine Learning Research 6*, 1705–1749.

BAR-HILLEL, A., HERTZ, T., SHENTAL, N., AND WEINSHALL, D. 2003. Learning distance functions using equivalence relations. In *Proceedings of ICML*. Washington, DC, 11–18.

BAR-HILLEL, A., HERTZ, T., SHENTAL, N., AND WEINSHALL, D. 2005. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research 6*.

BASU, S., BANERJEE, A., AND MOONEY, R. J. 2002. Semi-supervised clustering by seeding. In *Proceedings of ICML*. 19–26.

BASU, S., BANERJEE, A., AND MOONEY, R. J. 2004. Active semi-supervision for pairwise constrained clustering. In *Proceedings of SIAM SDM*.

BASU, S., BILENKO, M., AND MOONEY, R. J. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of ACM SIGKDD*. Seattle, WA, 59–68.

BASU, S., DAVIDSON, I., AND WAGSTAFF, K. 2008. *Clustering with Constraints: Algorithms, Applications and Theory*. Chapman & Hall/CRC Press Data Mining and Knowledge Discovery Series.

BIE, T. D., MOMMA, M., AND CRISTIANINI, N. 2003. Efficiently learning the metric using side-information. In *Proc. of the 14th International Conference on Algorithmic Learning Theory (ALT2003)*. Lecture Notes in Artificial Intelligence, vol. 2842. Springer, 175–189.

BILENKO, M. AND MOONEY, R. J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of ACM SIGKDD*. Washington, DC, 39–48.

BLAKE, C. L. AND MERZ, C. J. 1998. UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

BUNTINE, W. L. 1994. Operations for learning with graphical models. *Journal of Artificial Intelligence Research 2*, 159–225.

CHANG, H. AND YEUNG, D.-Y. 2004. Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*.

COHN, D., CARUANA, R., AND MCCALLUM, A. 2003. Semi-supervised clustering with user feedback. Tech. Rep. TR2003-1892, Cornell University.

DAVIDSON, I., BASU, S., AND WAGSTAFF, K. 2006. In *Proceedings of the Tenth European Principles and Practice of KDD (PKDD).*

DAVIDSON, I., ESTER, M., AND RAVI, S. S. 2007. Efficient incremental clustering with constraints. In *Proceedings of the Thirteen ACM Conference on Data Mining and Knowledge Discovery*.

DAVIDSON, I. AND RAVI, S. 2005a. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM-05)*.

DAVIDSON, I. AND RAVI, S. S. 2005b. Hierarchical clustering with constraints: Theory and practice. In *Proceedings of the Nineth European Principles and Practice of KDD (PKDD)*. 59–70.

DAVIDSON, I. AND RAVI, S. S. 2006. Identifying and generating easy sets of constraints for clustering. In *Proceedings of the* 21*st* *AAAI Conference*.

DAVIDSON, I. AND RAVI, S. S. 2007. Hierarchical clustering with constraints: Theory and practice. *Knowledge Discovery and Data Mining 14,* 1.

DEMIRIZ, A., BENNETT, K. P., AND EMBRECHTS, M. J. 1999. Semi-supervised clustering using genetic algorithms. In *Proceedings of ANNIE*. 809–814.

DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *JRSSB 39*, 1–38.

DHILLON, I. S., FAN, J., AND GUAN, Y. 2001. Efficient clustering of very large document collections. In *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers.

DHILLON, I. S. AND GUAN, Y. 2003. Information theoretic clustering of sparse co-occurrence data. In *Proceedings of ICDM*. 517–521.

EISEN, M. B., SPELLMAN, P. T., BROWN, P. O., AND BOTSTEIN, D. 1998. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences, USA 95*, 14863–14848.

ELKAN, C. 2003. Using the triangle inequality to accelerate *k*-means. In *ICML*.

FREUND, Y. AND SCHAPIRE, R. E. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*, L. Saitta, Ed. Morgan Kaufmann, 148–156.

FREUND, Y., SEUNG, H. S., SHAMIR, E., AND TISHBY, N. 1997. Selective sampling using the query by committee algorithm. *Machine Learning 28*, 133–168.

GONDEK, D. AND HOFMANN, T. 2004. Non-redundant data clustering. In *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE Computer Society, Washington, DC, USA, 75–82.

HERTZ, T., BAR-HILLEL, A., AND WEINSHALL, D. 2004. Boosting margin based distance functions for clustering. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*.

HOCHBAUM, D. S. AND SHMOYS, D. B. 1985. A best possible heuristic for the *k*-center problem. *Mathematics of Operations Research 10(2)*, 180–184.

HOFMANN, T. AND BUHMANN, J. M. 1998. Active data clustering. In *Advances in Neural Information Processing Systems 10*.

KEARNS, M., MANSOUR, Y., AND NG, A. Y. 1997. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proceedings of UAI*. 282–293.

KLEIN, D., KAMVAR, S. D., AND MANNING, C. D. 2002. 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*.

KLEINBERG, J. AND TARDOS, E. 1999. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *Proceedings of FOCS*. 14–23.

LANGE, T., LAW, M. H. C., JAIN, A. K., AND BUHMANN, J. M. 2005. Learning with constrained and unlabeled data. In *CVPR*. San Diego, CA, 731–738.

LAW, M. H. C., TOPCHY, A., AND JAIN, A. K. 2005. Model-based clustering with probabilistic constraints. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM-05)*.

LEWIS, D. AND GALE, W. 1994. A sequential algorithm for training text classifiers. In *Proceedings of Seventeenth International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94)*.

LU, Z. AND LEEN, T. K. 2005. Semi-supervised learning with penalized probabilistic clustering. In *Advances in Neural Information Processing Systems 17*.

MCCALLUM, A. AND NIGAM, K. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of ICML*. Madison, WI.

NANNI, M. 2005. Speeding-up hierarchical agglomerative clustering in presence of expensive metrics. In *PAKDD*.

NEAL, R. M. AND HINTON, G. E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, M. I. Jordan, Ed. MIT Press, 355–368.

PELLEG, D. AND BARAS, D. 2007. K-means with large and noisy constraint sets. In *ECML*.

RAND, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*, 366.

SEEGER, M. 2000. Learning with labeled and unlabeled data.

SEGAL, E., WANG, H., AND KOLLER, D. 2003. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics 19*, i264–i272.

SINKKONEN, J. AND KASKI, S. 2000. Semisupervised clustering based on conditional distributions in an auxiliary space. Tech. Rep. A60, Helsinki University of Technology.

WAGSTAFF, K. AND CARDIE, C. 2000. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, Palo Alto, CA, 1103–1110.

WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHROEDL, S. 2001a. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHROEDL, S. 2001b. Constrained K-Means clustering with background knowledge. In *Proceedings of ICML*. 577–584.

WAGSTAFF, K. L. 2002. Intelligent Clustering with Instance-Level Constraints. Ph.D. thesis, Cornell University.

XENARIOS, I., FERNANDEZ, E., SALWINSKI, L., DUAN, X. J., THOMPSON, M. J., MARCOTTE, E. M., AND EISENBERG, D. 2001. DIP: The database of interacting proteins: 2001 update. *Nucleic Acids Research 29,* 1, 239–241.

XING, E. P., NG, A. Y., JORDAN, M. I., AND RUSSELL, S. 2003. Distance metric learning, with application to clustering with side-information. In *NIPS 15*.

YAN, R., ZHANG, J., YANG, J., AND HAUPTMANN, A. G. 2004. A discriminative learning framework with pairwise constraints for video object classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

YANG, K., YANG, M., AND KAFATOS, M. 2001. A feasible method to find areas with constraints using hierarchical depth-first clustering. In *Scientific and Statistical Database Management Conference*.

ZAIANE, O., FOSS, A., LEE, C., AND WANG, W. 2000. On data clustering analysis: Scalability, constraints and validation. In *PAKDD*.

ZHU, X. 2005. Semi-supervised learning literature survey. Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

Received September 2007; accepted XXXX XXXX