H-0253 (H0706-012) June 18, 2007
Computer Science

IBM Research Report

K-means with Large and Noisy Constraint Sets

Dan Pelleg, Dorit Baras
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

K-means with Large and Noisy Constraint Sets

Dan Pelleg and Dorit Baras

IBM Haifa Labs
dpelleg@il.ibm.com doritb@il.ibm.com

Abstract. We focus on the problem of clustering with soft instance-level constraints. Recently, the CVQE algorithm was proposed in this context. It modifies the objective function of traditional K-means to include penalties for violated constraints. CVQE was shown to efficiently produce high-quality clustering of UCI data. In this work, we examine the properties of CVQE and propose a modification that results in a more intuitive objective function, with lower computational complexity. We present our extensive experimentation, which provides insight into CVQE and shows that our new variant can dramatically improve clustering quality while reducing run time. We show its superiority in a large-scale surveillance scenario with noisy constraints.

1 Introduction

Recently, a growing interest in utilizing side-information in clustering led to a variety of new clustering techniques. The side information is used to encode a tacit bias to counter that of the original clustering algorithm. In this sense, the new algorithm can be thought of as supervised. But in contrast to traditional supervision, the ground truth labels need not be explicitly present in the input.

Typically, the side information is in the form of pairwise instance-level constraints. Constraints of this type come in two flavors: a *must-link (ML)* constraint, to indicate that a pair of input points need to be in the same output cluster, and a *cannot-link (CL)* constraint, to indicate the opposite. These types of constraints were thoroughly investigated and have been shown to improve results in different application areas. Some of these areas include GPS lane finding [1], video and image indexing [2, 3], robot navigation [4], image segmentation [5], and text categorization [6, 7]. The constraints are considered to increase cluster purity, decrease convergence time, and reduce error [8].

The method in which the constraints are acquired depends on the application itself. For example, spatial or temporal proximity of observations may be used to induce constraints, or user feedback on a clustering result may be used in an active-learning or semi-supervised setting. In experimentation, it is also popular to use the ground truth labels to induce constraints.

In general, existing methods fall into one of two categories: constraint-driven and distance-driven. The first type tries to directly satisfy the constraints. There are hard and soft versions of these, which vary in their ability to ignore some

constraints. The second type learns a distance metric from the constraints, and it is later used in a constraint-agnostic clustering algorithm.

This work is motivated by a surveillance application. In this setting, a sensor (e.g., a video camera), or a network of such sensors, is located in a public area. The sensor can locate objects in a 2-D or 3-D space. It can also track their movement over a short period of time. For example, if object locations are recorded every minute, the sensor may also identify that the same object that was at location P at time t, is at location P' at time $t+1$. Noise in the measurements may come in the form of false tracking due to objects or agents leaving and entering the scene, occlusion, etc. The goal of the application is to perform long-term tracking of objects. That is, cluster the observation points such that each cluster corresponds to a single object. This naturally gives rise to the constrained clustering problem with the following characteristics:

- The number of data points and constraints is large (thousands or more, depending on the monitoring period and frequency).
- The constraints are mostly ML.
- The constraints are noisy.

This paper explores solutions to such problems. From the description above, some required properties for such solutions emerge: scalability, efficiency, and resilience to constraint noise. The latter immediately precludes hard satisfaction algorithms. Of the soft variants, algorithms based on K-means are a natural fit to the scalability requirement, since they are potentially linear in the number of points, dimensions, constraints and clusters.

We also note the common practice of augmenting the constraint set by transitive and entailed constraints. That is, if the input includes the constraints $ML(a,b)$ and $ML(b,c)$, then the transitive constraint $ML(a,c)$ is added in preprocessing. Similarly, the constraint set $\{ML(a,b), ML(d,e), CL(a,d)\}$ will entail the constraints $\{CL(a,e), CL(b,d), CL(b,e)\}$. This is a widely-used heuristic[9], but unfortunately, it cannot be used in our scenario for two reasons. First, the noise may introduce ML constraints between some members of different clusters. When the number of constraints is large, the probability of such an event is high, and the result would be the complete— and useless— clique in the ML graph. Second, the size of the augmented set is huge. In one experiment with around 20000 points and constraints, this kind of pre-processing generated approximately half a million constraints.

Davidson et al. [10] propose using a black-box method to evaluate the usefulness of constraints. Two measurements are defined on constraint sets: *informativeness* and *coherence*. Informativeness represents the tacit bias, due to the constraints, that is different from the algorithm's own bias. Coherence is the disparity between ML and CL pairs. These measures can be used to evaluate a given constraint set. In a convincing experiment, an extremely small constraint set is shown to dramatically enhance clustering results. Taking this idea further, the authors suggest using the same measures to filter constraint sets before feeding them to a constrained clustering algorithm. The benefit would be smaller and

cleaner sets, resulting in faster operation and increased accuracy. This approach seems like a viable alternative to our scalable algorithms. We look forward to the gap between this idea and a working embodiment to be bridged, enabling us to directly compare the two approaches.

The remainder of this paper is organized as follows. In Section 2 we describe the CVQE algorithm[4] on which we base our work, and examine its properties. In Section 3 we propose LCVQE (for "linear-time CVQE"), our variant of CVQE, and in Section 4 we present experimental results for UCI and tracking data.

2 The CVQE Algorithm

The unconstrained clustering problem is defined on instances $S_i, i = 1, ...n$ and a parameter K for the number of clusters. Let C_j be the centroid representing cluster j. Denote by Q_j the set of instances that are closest to C_j. The K-means algorithm uses the following update rule: $C_j = \frac{1}{|Q_j|} \sum_{s_i \in Q_j} s_i$, where after every centroid update, each instance is reassigned to the cluster of its closest centroid (i.e., the groups Q_j are recalculated). This update rule is derived from minimization of the vector quantization error function, $VQE = \frac{1}{2} \sum_{j=1}^{K} \sum_{s_i \in Q_j} \left(C_j - s_i\right)^2$.

The Constrained Vector Quantization Error(CVQE) algorithm[4] generalizes K-means to handle constraints. It does so by modifying the error function to penalize violated constraints. In the original notation, there are r must-link and s cannot-link constraints, $\left\{\left(s_1(i), s_2(i)\right)\right\}_{i=1}^{s+r}$. Let Q_j be the set of instances assigned to the j-th cluster, and C_j be the centroid corresponding to the j-th cluster. Define $M(x) = \{j \,|x \in Q_j\}$, and let $g(i) = M(s_1(i))$ and $g'(i) = M(s_2(i))$. Further, let $h(i)$ be the cluster index whose centroid is closest to C_i. As in Davidson et al. [4], in the case of violation, $s_2(i)$ is associated with the violation. Finally, $v(i)$ indicates whether the i-th constraint is violated. Namely, for $i = 1, ..., r$, $v(i) = 1 \leftrightarrow g(i) \neq g'(i)$ and for $i = r+1, ..., s+r$, $v(i) = 1 \leftrightarrow g(i) = g'(i)$, and $v(i) = 0$ in all other cases. The update rule is as follows:

$$C_j = \frac{1}{N_j} \Bigg\{ \sum_{s_i \in Q_j} (s_i) + \sum_{l=1, g(l)=j}^{r} v(l) \cdot C_{g'(l)} + \sum_{l=r+1, g(l)=j}^{s+r} v(l) \cdot C_{h(g'(l))} \Bigg\} \tag{1}$$

And $N_j = |Q_j| + \sum_{l=1, g(l)=j}^{r+s} v(l)$. Intuitively, in violations of ML constraints, one of the two affected centroids is moved towards the other, wheres CL violations move one of the points towards its next-closest centroid. Similarly to K-means, after each iteration, each instance is reassigned to minimize the error function (described below). Hence unlike K-means, Q_j can contain instances where C_j is not their closest centroid. This update rule minimizes the following error function $CVQE = \sum_{j=1}^{K} CVQE_j$, where

$$CVQE_j = \frac{1}{2}\sum_{s_i \in Q_j} T_{j,1} + \frac{1}{2}\sum_{l=1,g(l)=j}^{r} T_{j,2} + \frac{1}{2}\sum_{l=r+1,g(l)=j}^{r+s} T_{j,3} \qquad (2)$$

where $T_{j,1} = \left(C_j - s_i\right)^2$, $T_{j,2} = \left[\left(C_j - C_{g'(l)}\right)^2 \cdot v(l)\right]$, and $T_{j,3} = \left[\left(C_j - C_{h(g'(l))}\right)^2 \cdot v(l)\right]$.

In each step of the CVQE algorithm, each pair of instances that form a constraint are assigned such that the CVQE error function is minimized. The rest of the algorithm (initialization and termination) is the same as K-means.

We now discuss some properties of CVQE. First, the order of the points in a constraint is significant. Consider a violated constraint generated by the instances $(s_1(l), s_2(l))$ such that $s_1(l) \in Q_{g(l)}, s_2(l) \in Q_{g'(l)}$. Only $C_{g(l)}$ is affected by violation of this link, while $C_{g'(l)}$ is not affected at all. This observation holds in both the ML and CL cases.

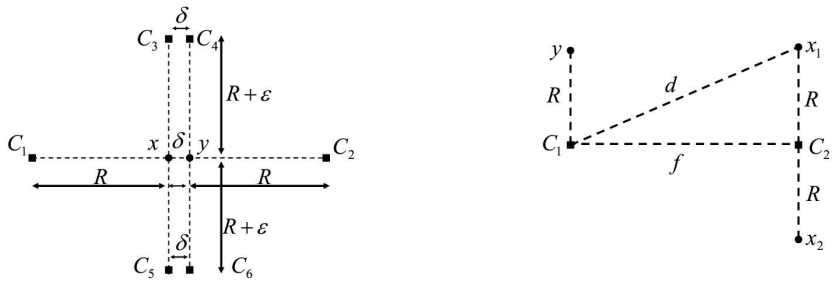
Second, determining the assignment that minimizes the error function requires $O(K^2)$ calculations for every constraint, which can be expensive when dealing with large numbers of either constraints or clusters. It is also not possible to prune out any possibility (other than the trivial $s_1(l) \in Q_{g'(l)}, s_2(l) \in Q_{g(l)}$) from the calculation. To see this, consider Figure 1(a). The pair (x, y) is ML and the current centroids are $C_i, i = 1, ..., 6$. The distances are shown in the figure. Depending on the values of R, δ, and ϵ, points may be assigned to any of the pairs (C_1, C_2), (C_3, C_4), or (C_5, C_6). Hence, all K^2 options must be checked for every constraint.

Another issue arises from the fact that the penalty for violated links depends on the distance between the corresponding centroids, but the locations of the instances are not taken into account. Consider the two clustering problems shown in Figure 1(b). In both, the initial centroids are C_1, C_2. One problem includes the ML (x_1, y), while the other includes ML (x_2, y).

Table 1 summarizes the available assignments in each case and the CVQE value. Note that, regardless of d and f, both problems always have the same solution, although our intuition says that violating (x_1, y) is "worse" than violating (x_2, y). Furthermore, the corrective action in both cases is the same: move C_1 along the line connecting it to C_2, completely ignoring the relative orientation of the offending instance.

Table 1. CVQE values for Figure 1(b)

Constraint	$y \in Q_1, x_i \in Q_2$	$x_i, y \in Q_1$	$x_i, y \in Q_2$
(x_1, y)	$R^2 + R^2 + f^2$	$R^2 + d^2$	$R^2 + d^2$
(x_2, y)	$R^2 + R^2 + f^2$	$R^2 + d^2$	$R^2 + d^2$



(a) (b)

Fig. 1. CVQE examples

3 The LCVQE Algorithm

Our modification of CVQE follows. The proposed algorithm minimizes a target function composed of the vector quantization error as well as a penalty for violated constraints, in the style of CVQE. For each constraint assignment, the LCVQE algorithm considers at most two naturally chosen clusters, hence its complexity is independent of K. Informally, violated ML constraints update each centroid toward the opposite instance, hence they are symmetric in instance order. For violated CL constraints, the instance that is far from the cluster centroid is determined and the closest centroid to that instance (other than the current centroid) is moved towards it.

To formalize the algorithm, we define two new functions. $R_j(l)$ returns the instance among $s_1(l), s_2(l)$ whose distance to C_j is larger. MM(s) returns the centroid which is the closest to s, other than $C_{M(s)}$. The LCVQE update rule is given by:

$$C_j = \frac{1}{N_j}\Bigg\{ \sum_{s_i \in Q_j} s_i + \frac{1}{2} \sum_{l=1, g(l)=j}^{r} v(l) \cdot s_2(l) + \frac{1}{2} \sum_{l=1, g'(l)=j}^{r} v(l) \cdot s_1(l) \qquad (3)$$
$$+ \sum_{l=r+1, j=\mathrm{MM}(R_{M(s_1(l))}(l))}^{s+r} v(l) \cdot R_{M(s_1(l))}(l) \Bigg\}$$

$N_j = |Q_j| + \frac{1}{2}\sum_{l=1,g(l)=j}^{r} v(l) + \frac{1}{2}\sum_{l=1,g'(l)=j}^{r} v(l) + \sum_{l=r+1,j=\mathrm{MM}(R_{M(s_1(l))}(l))}^{s+r} v(l)$.
This update rule minimizes the error function:

$$E_j = \frac{1}{2}\sum_{s_i \in Q_j} T_{j,1} + \frac{1}{2}\sum_{l=1,g(l)=j}^{r} T_{j,2} + \frac{1}{2}\sum_{l=1,g'(l)=j}^{r} T_{j,3} + \frac{1}{2}\sum_{l=r+1,j=\mathrm{MM}(R_{M(s_1(l))}(l))}^{s+r} T_{j,4}$$

Here,

$T_{j,1} = \left(C_j - s_i\right)^2$ $\quad$ $T_{j,2} = \left[\frac{1}{2}\left(C_j - s_2(l)\right)^2 \cdot v(l)\right]$

$T_{j,3} = \left[\frac{1}{2}\left(C_j - s_1(l)\right)^2 \cdot v(l)\right]$ $\quad$ $T_{j,4} = \left[\left(C_j - R_{M(s_1(l))}(l)\right)\right)^2 \cdot v(l)\right]$

The corresponding pseudo-code is:

1. $\mathrm{GMLV}_j = \{\}, \mathrm{GCLV}_j = \{\}, \forall j \in \{1, ..., K\}$
2. Assign Q_j by the closet-centroid rule.
3. For each must-link $\{s_1(l), s_2(l)\}$, such that C_j is the closest centroid to $s_1(l)$ and C_n is the closest centroid to $s_2(l)$, compute the following quantities:
   (a) $\frac{1}{2}\left[\left(s_1(l) - C_j\right)^2 + \left(s_2(l) - C_n\right)^2\right] + \frac{1}{4}\left[\left(s_1(l) - C_n\right)^2 + \left(s_2(l) - C_j\right)^2\right]$
   (b) $\frac{1}{2}\left(s_1(l) - C_j\right)^2 + \frac{1}{2}\left(s_2(l) - C_j\right)^2$
   (c) $\frac{1}{2}\left(s_1(l) - C_n\right)^2 + \frac{1}{2}\left(s_2(l) - C_n\right)^2$
   If (a) is minimal, $\mathrm{GMLV}_j = \mathrm{GMLV}_j \cup s_2(l)$ and $\mathrm{GMLV}_n = \mathrm{GMLV}_n \cup s_1(l)$, assign s_1 to Q_j and s_2 to Q_n.
   If (b) is minimal, assign s_1 and s_2 to Q_j.
   If (c) is minimal, assign s_1 and s_2 to Q_n.
4. For each cannot-link $\{s_1(l), s_2(l)\}$, such that C_j is the closest centroid to $s_1(l)$ and C_n is the closest centroid to $s_2(l)$, let $R_n(l) = \arg\max_{s_i(l), i=1,2}\left(s_i(l) - C_n\right)^2$. Let $j = MM(R_n(l))$, be the index of the centroid that is closest to $R_n(l)$ other than n. Compute the following quantities:
   (a) $\frac{1}{2}\left(s_1(l) - C_j\right)^2 + \frac{1}{2}\left(s_2(l) - C_j\right)^2 + \frac{1}{2}\left(R_j(l) - C_{\mathrm{MM}(R_j(l))}\right)^2$
   (b) $\frac{1}{2}\left(s_1(l) - C_n\right)^2 + \frac{1}{2}\left(s_2(l) - C_n\right)^2 + \frac{1}{2}\left(R_n(l) - C_{\mathrm{MM}(R_n(l))}\right)^2$
   (c) $\frac{1}{2}\left[\left(s_1(l) - C_j\right)^2 + \left(s_2(l) - C_n\right)^2\right]$
   If (a) is minimal, $\mathrm{GCLV}_{\mathrm{MM}(R_j(l))} = \mathrm{GCLV}_{\mathrm{MM}(R_j(l))} \cup R_j(l)$, assign s_1 and s_2 to Q_j.
   If (b) is minimal, $\mathrm{GCLV}_{\mathrm{MM}(R_n(l))} = \mathrm{GCLV}_{\mathrm{MM}(R_n(l))} \cup R_n(l)$, assign s_1 and s_2 to Q_n.
   If (c) is minimal, assign s_1 to Q_j and s_2 to Q_n.
5. Update C_j as follows:

$$C_j = \frac{1}{N_j}\left\{\sum_{s_i \in Q_j} s_i + \frac{1}{2}\sum_{s_i \in \mathrm{GMLV}_j} s_i + \sum_{s_i \in \mathrm{GCLV}_j} s_i\right\} \tag{4}$$

6. Repeat steps $1-5$ till convergence.

Here Q_j is the set of instances assigned to the jth cluster and $N_j = |Q_j| + \frac{1}{2}|GMLV_j| + |GCLV_j|$. The convergence criterion can be near-quiescence of cluster movement, cluster assignment, or total LCVQE (our implementation choice).

The LCVQE algorithm requires $O(d)$ operations (for d dimensions) in each step because only three[1] possible assignments are checked, regardless of K. Hence, this algorithm is faster and efficient for problems having large number of constraints or clusters.

Another benefit of the algorithm is that the constraints are symmetric and that the centroid that is updated depends on the exact setting of both instances rather than the violating instance alone.

Finally, our algorithm does a better job of handling the example shown in Figure 1(b). Table 2 summarizes the available assignments in each case and the LCVQE values. Assume that the assignment in both problems is $x_i \in Q_2, y \in Q_1$. In the case of (x_1, y), the centroid is updated as follows: $C_1 = (y + \frac{1}{2}x_1)/1.5$. In the case of (x_2, y), the centroid is updated as follows: $C_1 = (y + \frac{1}{2}x_2)/1.5$, which is intuitively better (because the centroid is moving toward the mean of the instances rather then toward the other centroid).

Table 2. CVQE values for Figure 1(b)

Must-link	$y \in Q_1, x_i \in Q_2$	$x_i, y \in Q_1$	$x_i, y \in Q_2$
(x_1, y)	$(d^2 + d^2)/2$	d^2	d^2
(x_2, y)	$(d^2 + d^2)/2$	d^2	d^2

4 Experiments

We first compare the performance of LCVQE and CVQE on UCI data. We implemented both, as well as K-means, in C, and used a $3.6GHz$ Pentium 4 machine for testing. Times in the plots are all in seconds.

The data sets and their properties are described in Table 3. We drew random data pairs to generate constraints. Noise was inserted with probability $p = 1\%$ by changing the labels in a pair to labels drawn uniformly from the set of classes. Afterwards, the labels were compared to generate either an ML or a CL constraint. In each case 25 ML and 25 CL constraints were generated.

We generated the augmented set of transitive and entailed constraints in pre-processing, which did not contribute towards the measured run time of any algorithm. To measure clustering performance, we used NMI[7]. Let $H(\cdot)$ denote the Shannon entropy and $I(X;Y)$ the mutual information of variables X and Y. Then NMI is defined as: $\frac{I(C;K)}{(H(C)+H(K))/2}$, where C denotes the ground truth labels and K is the output of some clustering algorithm.

[1] The fourth option can be trivially shown to be redundant.

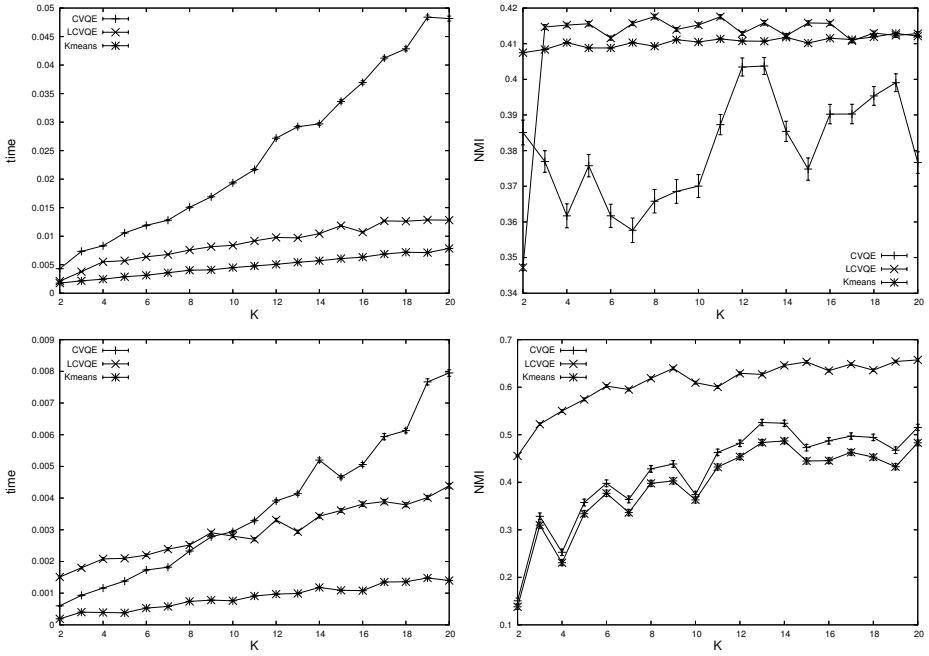


Fig. 2. Performance on UCI data, Wine and Iris

Figures 2 through 5 show NMI and run time values averaged over 100 runs. Accuracy is increased dramatically for all values of K in the Iris and Glass datasets, and is significantly better for most values of K for Ionosphere, Wine, and Pima, and at par for E-coli and Breast. Note that for the Glass and Wine datasets, CVQE is substantially worse than unconstrained K-means, a phenomenon we did not observe for LCVQE.

For run times, the quadratic growth in K is clearly visible for CVQE, whereas LCVQE (and, as expected, K-means), are linear in K. At the same time, there are cases where LCVQE is slower, especially at the low end of K. We explore this point below.

We also repeated the experiments, with $p = 5\%$ and $p = 10\%$, both with 50 and 75 constraints. They are similar to the results ones seen above. We show representative results for the Iris dataset in Figure 6.

Recall that, for each constraint, LCVQE searches over a space that includes just two centroids, whereas CVQE performs the search over all K centroids. This reduction in search space movement has the potential to make each change in centroids smaller. To test this hypothesis, we measured the average number of iterations to convergence for both versions, as well as the total centroid movement per iteration. (See Table 4). Additionally, we counted the average number

Table 3. Data sets used in the experiments

Name	Points	Dimensions	Classes
Iris	150	4	3
Breast	682	9	2
Wine	178	13	3
E-coli	336	8	8
Pima	768	8	2
Glass	214	9	7
Ionosphere	351	33	2

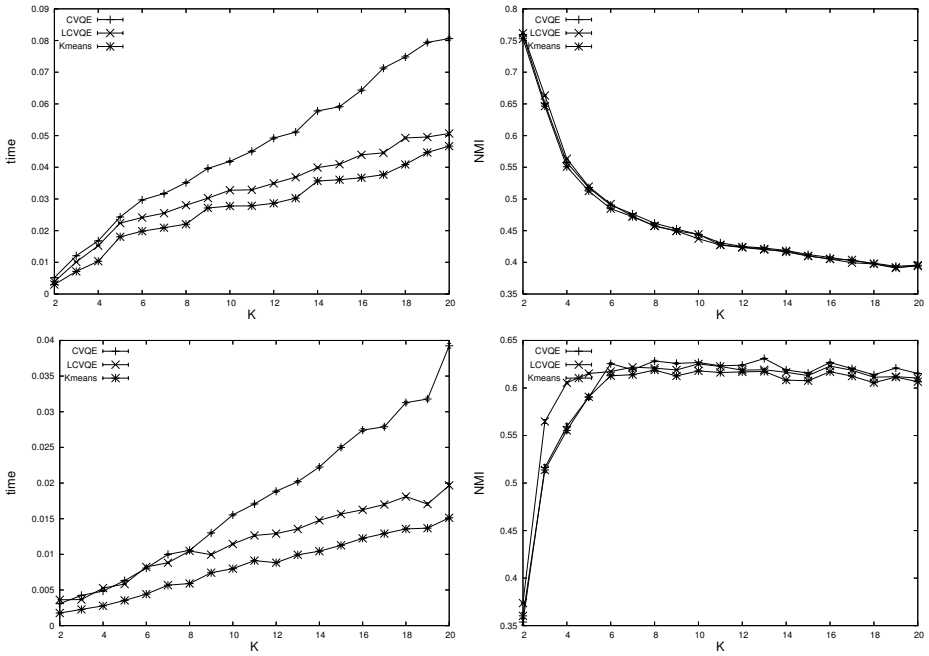


Fig. 3. Performance on UCI data, Breast cancer and E-coli

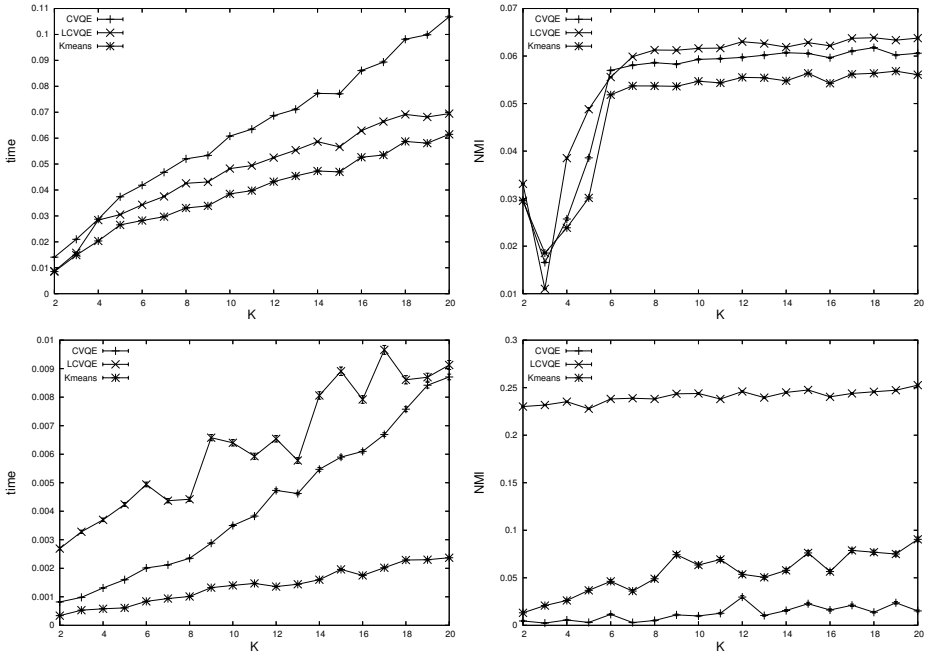


Fig. 4. Performance on UCI data, Pima and Glass

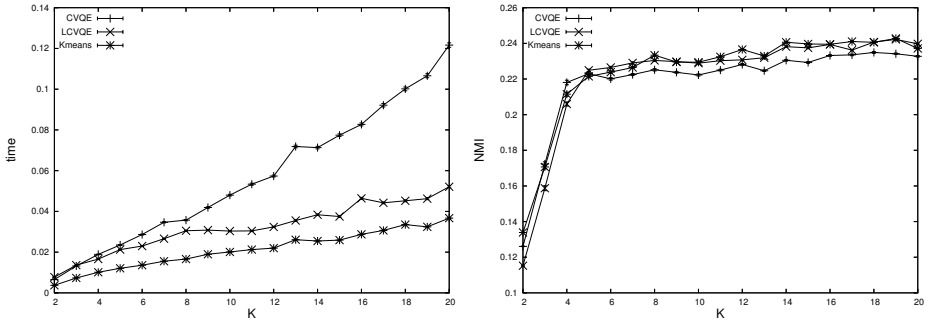


Fig. 5. Performance on UCI data, Ionosphere.

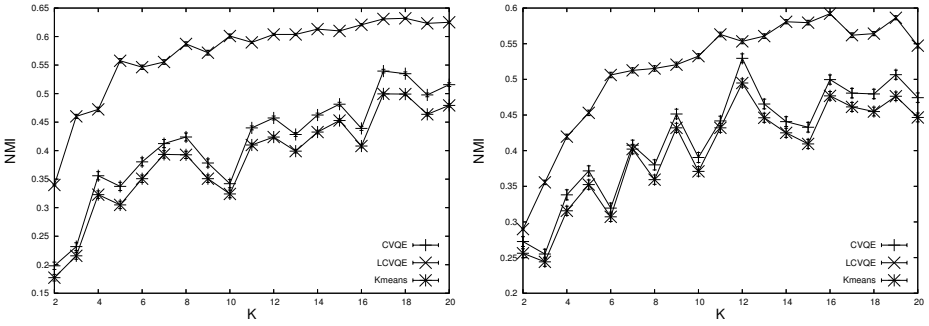


Fig. 6. Performance on Iris, 75 constraints, $p = 1\%$ (left) and $p = 10\%$ (right).

Table 4. Low-level performance measures for $k = 10$.

Name	Iterations		Movement		Clusters	
	LCVQE	CVQE	LCVQE	CVQE	LCVQE	CVQE
Iris	9.8±0.2	4.4±0	2.9±0	2.4±0	2.7±0	1.7±0
Breast	20.2±0.1	20.6±0.1	6.4±0	6.1±0	8.5±0	8.3±0
Wine	14.5±0.2	15.3±0.1	258.5±1.7	157.4±1.2	3.6±0	2.61±0
E-coli	14.9±0.1	12.9±0.1	0.46±0	0.47±0	6.1±0	5.8±0
Pima	27.6±0	29±0	77.7±0	72.3±0	7.9±0	7.7±0
Glass	11.1±0.2	3.3±0	23.4±0.2	28.4±0.2	2.3±0	1.1±0
Ionosphere	16.1±0.2	14.8±0.1	3.5±0	3.6±0	9.8±0	9.8±0

of unique labels output by each algorithm[2]. We see that LCVQE does tend to iterate longer than CVQE, which explains the longer run times for low values of K. We also see that CVQE is fairly likely to orphan many of the centroids, resulting in solutions with inherently poor quality.

We also examined the dependence of run time on the size of the input. Here we added to the mix the MPC-Kmeans algorithm[11], which is an efficient hybrid of the distance-learning and constraint-satisfaction approaches[3]. We first varied the number of constraints. The data used was 100 random lines from the "cover type" UCI data set (54 dimensions, 7 classes), re-drawn at every run. Constraint noise was added as above. See Figure 7. While the exact values are not comparable (MPC-Kmeans is implemented in Java), the trends are valid. We see that the K-means variants are linear in the number of constraints, whereas MPC-Kmeans is super-linear. Here, too, the entailed constraints were

[2] Both algorithms were configured to not delete empty centroids. Even if they exist, the final assignment may or may not make use of them, resulting in cases with less than K unique labels in the output.

[3] In terms of NMI, it outperforms LCVQE on the small UCI datasets.

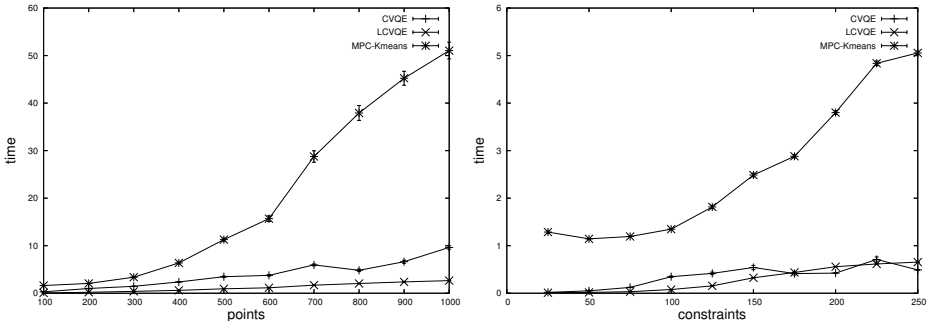


Fig. 7. Run time vs. number of constraints and points, $k = 5$. Results are averages over 30 runs.

generated by the wrapper script, and MPC-Kmeans was instructed not to run its own entailment code.

All algorithms are linear in the number of points (data not shown). But in our motivating example, the number of constraints is generally linear in the number of points. We therefore conducted another experiment with the cover type data, this time generating as many constraints as there are data points. (See Figure 7). Again we see the super-linear behavior of MPC-Kmeans, making it unsuitable for large-scale inputs.

Our final experiment tests performance on surveillance data from a realistic arena. In it, ten uncontrolled agents move in a 2-D space and interact among themselves and with the surroundings. Each minute, a snapshot is taken, and the location and true identity of each agent are recorded. We generate raw data that corresponds to the locations of agents, and ML constraints between each agent's location and its location at the previous time step. In a sense, this is a version of the famous GPS lane finding data but with greater freedom of movement for the agents. Noise was added as above. Here we did not augment the constraint set (nor did MPC-Kmeans run its own augmentation routine). Because of the long chains in the ML graph, the transitive closure becomes huge and mostly uninformative. (See Figure 8). We observe that LCVQE produces better or equivalent results to CVQE, while running much faster. In particular, the performance of MPC-Kmeans degrades very quickly with the number of constraints increases. We can speculate that this could be the effect of noise. Another possible reason is the absence of the connected-component heuristic — published work on MPC-Kmeans did not explore any of these scenarios.

5 Conclusion

Having emerged as a new technology a few short years ago, constraint clustering methods are now transitioning to the status of established practice. Consequently, the question of constraint acquisition is increasing in importance. Without automatic constraint generation, plugging in a constrained method in

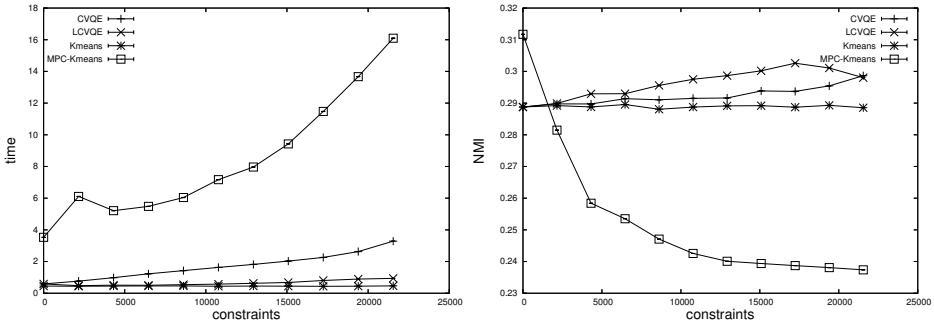


Fig. 8. Performance on tracking data, $k = 10$.

place of a traditional unsupervised method is impossible. We hypothesize that, in many realistic scenarios, the graphs of generated constraints will look very different from the label-based constraint sets that are traditionally used to evaluate new algorithms. In particular, the graphs of ML constraints are likely to contain long chains rather than (dense or sparse) cliques. An interesting avenue of research is to explore the effect of graph structure and noise on the qualities and desired properties of constrained clustering algorithms, in the spirit of Davidson et al.[12]. We explore this issue in a forthcoming paper.

In this light, we discuss a real-world tracking scenario where data points and constraints are plentiful and possibly noisy. This notion alone breaks the consistency assumption central to many of the existing constrained clustering algorithms, resulting in poor performance. We propose a scalable and robust algorithm, based on the CVQE framework, capable of operating in this kind of environment. We show extensive experimental results that shed light on the relative performance of both algorithms and compare them to a distance-learning algorithm.

Acknowledgments

We thank Ian Davidson for helpful comments and discussion.

References

1. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In Brodley, C., Danyluk, A., eds.: Proceeding of the 17th International Conference on Machine Learning, San Francisco, CA, Morgan Kaufmann (2001)
2. Lin, W.H., Hauptmann, A.: Structuring continuous video recordings of everyday life using time-constrained clustering. In: IS&T/SPIE Symposium on Electronic Imaging, San Jose, CA (January 2006)

3. Hertz, T., Shental, N., Bar-Hillel, A., Weinshall, D.: Enhancing image and video retrieval: Learning via equivalence constraints. In: In Proc. of IEEE Conference on Computer Vision and Pattern Recognition. (2003)
4. Davidson, I., Ravi, S.S.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: 5th SIAM Data Mining Conference. (2005)
5. Yu, S.X., Shi, J.: Grouping with directed relationships. Lecture Notes in Computer Science **2134** (2001)
6. Cohn, D., Caruana, R., McCallum, A.: Semi-supervised clustering with user feedback. Technical report, Cornell University (2003) TR2003-1892.
7. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W., eds.: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, ACM (August 2004) 59–68
8. Basu, S., Davidson, I.: Clustering with constraints: Theory and practice. Online Proceedings of a KDD tutorial (2006) `http://www.ai.sri.com/~basu/kdd-tutorial-2006/`.
9. Davidson, I., Ravi, S.S.: Identifying and generating easy sets of constraints for clustering. In: AAAI, AAAI Press (2006)
10. Davidson, I., Wagstaff, K., Basu, S.: Measuring constraint-set utility for partitional clustering algorithms. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: PKDD. Volume 4213 of Lecture Notes in Computer Science., Springer (2006) 115–126
11. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In Brodley, C.E., ed.: ICML, ACM (2004)
12. Davidson, I., Ravi, S.S.: Intractability and clustering with constraints. In: ICML. (2007)