Vol. 2 Issue 2, Feb.2012, pp. 257-262

Adding Domain Knowledge In Terms Of Pair-wise Instance Level Constraints, In Clustering, To Refine Road Map

Prarthana A. Deshkar, Dr. Manali Kshirsagar

(Dept. of Computer Technology, Yashwantrao Chavan Collage of Engineering, Nagpur, Maharashtra India)
(Head Dept. of Computer Technology, Yashwantrao Chavan Collage of Engineering, Nagpur, Maharashtra India)

Abstract:

Clustering algorithms, which automatically divide a data set into meaningful sub-groups, have been especially successful in both areas, in the corporate world to extract useful information from large database, as well as in academic research. In some cases information about the problem domain is available in addition to the data instances themselves. This information can be added in the clustering algorithm to make the algorithm domain specific. Domain knowledge should be added in algorithm because, any clustering algorithm can detect general trends and patterns in data, but they cannot make use of additional knowledge specific to the problem at hand, as a human expert can. In this paper we show how instance level constraints are added to make general K-Means algorithm more intelligent. For implementation purpose we have taken the GPS traces, which after implementation can represent road lanes.

Keywords- **Constraints, Domain knowledge, Instance level, Lane finding problem, Unsupervised learning.**

I. Introduction

Unsupervised learning algorithms have had some impressive successes, like clustering algorithms, which automatically divide a data set into meaningful sub-groups. Clustering algorithms are unsupervised in that they do not have access to any specific information about what they should be seeking. Despite the many applications of clustering algorithms, their very generality also limits their performance on any specific task. The traditional alternative to unsupervised learning has been supervised learning, which adapts to the problem at hand by not only exploiting but requiring problem-specific knowledge. But supervised learning is an all-or-nothing paradigm; it cannot make use of any unlabeled data. So to get rid of these extremities, the body of hybrid algorithms which seek to limit the amount of manual labeling that must be done yet still make use of domain-specific information when it is available, is extended to have 'Intelligent clustering' [1]. This paper evaluates this intelligent clustering algorithm that can take advantage of problem-specific information to become a temporary expert in the domain at hand. In particular, these "intelligent clustering" methods make use of information expressed as constraints between two items in the data set. These constraints are called as the instance level constraints. It places restrictions on individual pairs of items with regards to their relative cluster membership. So the resulting intelligent clustering method will be able to accept domain-specific information in the form of constraints on the output clusters.

At the most general level, each constraint is an instance-level statement about a pair of items in the data set that indicates a preference for being placed into the same cluster, or, alternatively, into different clusters.

II. Instance level constraints

Instance – level constraints place restrictions on individual pairs of items with regards to their relative cluster membership. They may take the form of partial labels on the data set, user feedback in interactive systems, or direct constraints on the relative placement of item pairs. To have more elaboration, let us discuss one example on pair wise instance level constraints, in which we will take census data,

Table 1. Census Data

Name	**SSN**	**Age**	**Gender**	**Occupation**	**Income**
Riya	111-11-1111	30	F	Professional Dancer	$50,000
Pankaj	444-44-4444	30	M	Professional Dancer	$50,000
Neha	222-22-2222	29	F	Rocket Scientist	$90,000
Dhanush	777-77-7777	28	M	Assistant professor	$70,000

Now we will consider a simple binary function which will indicate marital relationship,

Table 2. Marital Relation

Pankaj	**Riya**
Dhanush	Neha
….	…..

This relation can be converted to form set of hard, instance - level constraints by creating must-link constraint for each married pair. The set of constraints presented to the intelligent clustering algorithm would then be, {Gary $=_m$ Diana, Tim $=_m$ Lyra, . . . }. When enforced, these constraints will guarantee that a married pair is never split across two clusters. In this way for any domain constrains can be added to have accurate results.

So In the context of partitioning algorithms, instance level constraints are a useful way to express a priori knowledge about which instances should or should not be grouped together. Consequently, we consider two types of pair wise constraints:

- **Must-link** constraints specify that two instances have to be in the same cluster.
- **Cannot-link** constraints specify that two instances must not be placed in the same cluster.

The must-link constraints denote a transitive binary relation over the instances. Consequently, when making use of a set of constraints (of both kinds), we take a transitive closure over the constraints. The full set of derived constraints is then presented to the clustering algorithm. In general constraints may be derived from the background knowledge about the domain or data set. So the general architecture of such constrained clustering algorithm will be like,

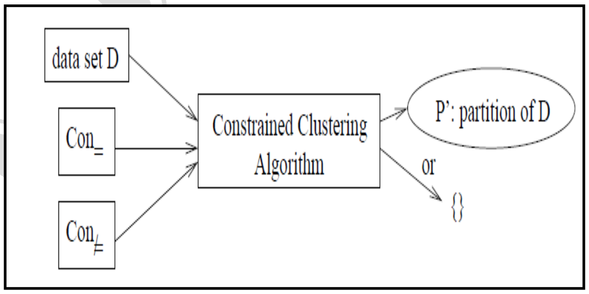


Fig. 1 Architecture of constrained clustering

III. Implementing constraints

A. Lane Finding In Digital Road Maps

Digital road maps currently exist that are used in several applications, such as generating personalized driving directions. However, these maps contain only coarse information about the location of a road. Map accuracy, in terms of how close the map points are to the true location of the road, is low. Our goal in this application is to refine digital maps by improving their accuracy and enhancing the fine details. We will produce maps that are annotated not only with the location of the road, but also with the location of individual road lanes. Approach to this problem is based on the observation that drivers tend to drive within lane boundaries (rather than, for example, straddling two lanes). Over time, lanes should correspond to “densely traveled” regions, in contrast to the lane boundaries, which should be “sparsely traveled.” Consequently, we hypothesized that it would be possible to collect data about the location of cars as they drive along a given road, and then cluster that data to automatically determine where the individual lanes are located.

B. Cluster Center Formation

Before implementing the modified clustering algorithm, which is given in table 1, some basic concepts such as cluster center calculation and average calculation, are to be reconstructed .To better analyze performance in this domain, we modified the cluster center representation. The usual way to compute the center of a cluster is to average all of its constituent points. There are two significant drawbacks of this representation. First, the center of a lane is a point halfway along its extent, which commonly means that points inside the lane but at the far ends of the road appear to be extremely far from the cluster center. Second, applications that make use of the clustering results need more than a single point to define a lane. Consequently, we instead represented each lane cluster with a line segment parallel to the centerline. This more accurately models what we conceptualize as \the center of the lane", provides a better basis for measuring the distance from a point to its lane cluster center, and provides useful output for other applications.

We express this lane representation with a tuple <l, r, y>, [2] which indicates the x coordinates of the left (l) and right (r) ends of the lane as well as the (constant) centerline offset of the lane, y:

$$Ct_i.l = \min d.x \quad (1)$$

$$Ct_i.r = \max d.x \quad (2)$$

$$Ct_i.y = 1/C_i.\sum d.y \quad (3)$$

Thus, the center of the cluster is a line segment from (Cti.l,Cti.y) to (Cti.r,Cti.y). This formulation more accurately models what we conceptualize as “the center of the lane,” provides a better basis for measuring the distance from a point to its lane cluster center, and also provides useful output for other applications.

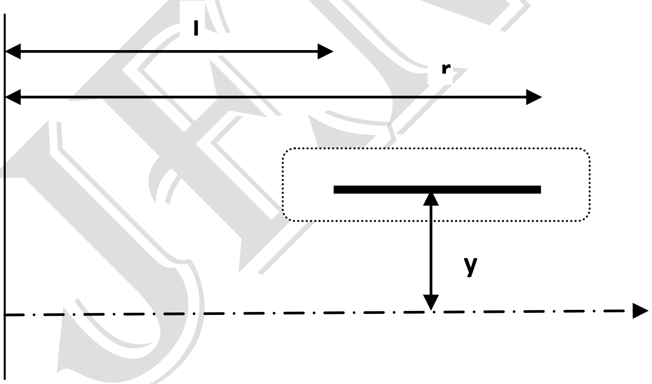


Fig. 2 cluster center representation

Due to our modification to the cluster center representation, we will make use of the following method for calculating distance [2]:

$$Dist(d, C_i) = sqrt((d.x - Cti.l)^2 + (d.y - Cti.y)^2) \quad (4)$$

or

$$Dist(d, C_i) = sqrt((d.x - Cti.r)^2 + (d.y - Cti.y)^2) \quad (5)$$

or

$$d.y - Cti.y \quad (6)$$

C. Generating lane Finding Constraints

The constraint types defined as must link and cannot link, provide us with a language to express our domain knowledge about the problem of lane finding. In this case, we focus on two domain-specific heuristics for generating constraints: trace contiguity and maximum separation.

Heuristic 1: Trace contiguity means that, in the absence of lane changes, all points a, b generated by the same vehicle in a single pass over a road segment should end up in the same lane. In other words, traces are contiguous in that all of the points from a single trace are linked. We convert this heuristic into a set of constraints in the following way:

For all a,b a.traverse = b.traverse, implies a,b belongs to must link constraint.

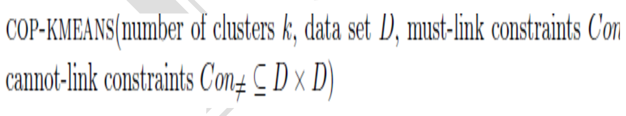
Heuristic 2: Maximum separation refers to a limit on how far apart two points can be (perpendicular to the centerline) while still being in the same lane. If two points are within ten meters of each other in distance along the road segment, and their centerline offsets differ by at least four meters, then we generate a constraint that will prevent those two points from being placed in the same cluster:

For all a,b |a.x − b.x| < 10.0 and |a.y − b.y| > 4.0), then (a, b) belongs to cannot link constraint.

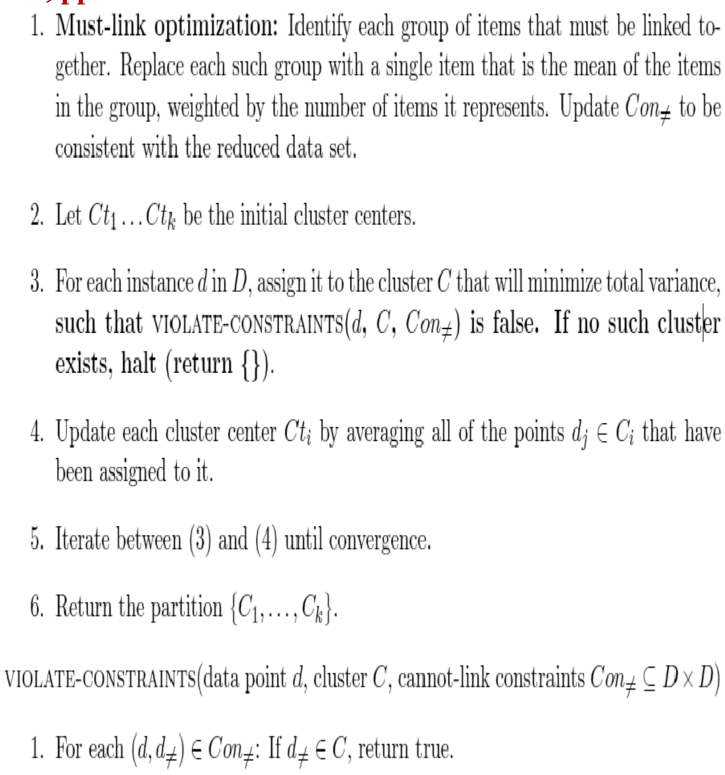
D. The Constrained Algorithm

The new modified intelligent clustering algorithm, COP-KMEANS [1] is as shown in table 1.

Table 3. Constrained clustering algorithm



COP-KMEANS(number of clusters k, data set D, must-link constraints Con
cannot-link constraints $Con_{\neq} \subseteq D \times D$)



```
1. Must-link optimization: Identify each group of items that must be linked together. Replace each such group with a single item that is the mean of the items in the group, weighted by the number of items it represents. Update $Con_{\neq}$ to be consistent with the reduced data set.

2. Let $Ct_1 \ldots Ct_k$ be the initial cluster centers.

3. For each instance $d$ in $D$, assign it to the cluster $C$ that will minimize total variance, such that VIOLATE-CONSTRAINTS($d$, $C$, $Con_{\neq}$) is false. If no such cluster exists, halt (return $\{\}$).

4. Update each cluster center $Ct_i$ by averaging all of the points $d_j \in C_i$ that have been assigned to it.

5. Iterate between (3) and (4) until convergence.

6. Return the partition $\{C_1, \ldots, C_k\}$.

VIOLATE-CONSTRAINTS(data point $d$, cluster $C$, cannot-link constraints $Con_{\neq} \subseteq D \times D$)

1. For each $(d, d_{\neq}) \in Con_{\neq}$: If $d_{\neq} \in C$, return true.
```

The algorithm takes in a data set (D), a set of must-link constraints (Con=), and a set of cannot-link constraints (Con≠). It returns a partition of the instances in D that satisfies all specified constraints.

The major modification is that, when updating cluster assignments, we ensure that none of the specified constraints are violated. We attempt to assign each point di to its closest cluster Cj . This will succeed unless a constraint would be violated. If there is another point d= that must be assigned to the same cluster as d, but that is already in some other cluster, or there is another point d= that cannot be grouped with d but is already in C, then d cannot be placed in C. We continue down the sorted list of clusters until we find one that can legally host d. Constraints are never broken; if a legal cluster cannot be found for d, the empty partition ({ }) is returned.

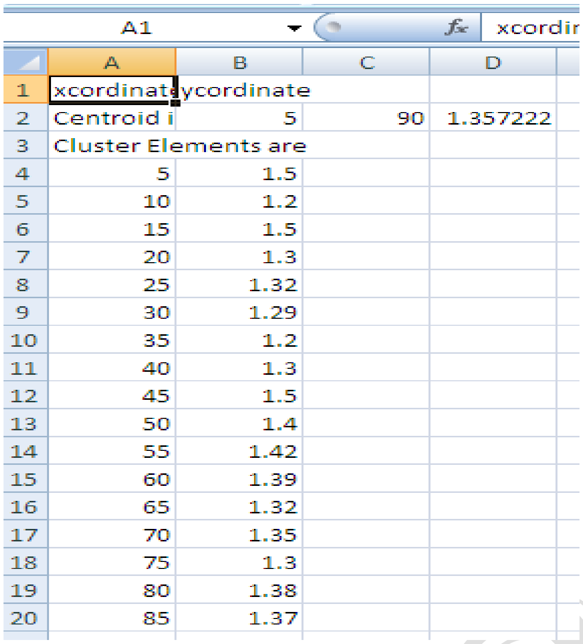
IV Experimental Observations

After applying the second heuristic to the data in hand, we are successful in getting the better results than the results which are got after applying simple K-Means algorithm.

[1] Although only the must-link constraints are transitive, the closure is performed over both kinds because, e.g, if di must link to dj which cannot link to dk, then we also know that di cannot link to dk.

Here the data we have taken is having two lanes.

Snapshot of output data which we got after applying the constrained K-Means algorithm is,



	A	B	C	D
1	xcordinat	ycordinate		
2	Centroid i	5	90	1.357222
3	Cluster Elements are			
4	5	1.5		
5	10	1.2		
6	15	1.5		
7	20	1.3		
8	25	1.32		
9	30	1.29		
10	35	1.2		
11	40	1.3		
12	45	1.5		
13	50	1.4		
14	55	1.42		
15	60	1.39		
16	65	1.32		
17	70	1.35		
18	75	1.3		
19	80	1.38		
20	85	1.37		

Fig. 3 Output data of Cluster I

The graphical representation will more clear the picture.

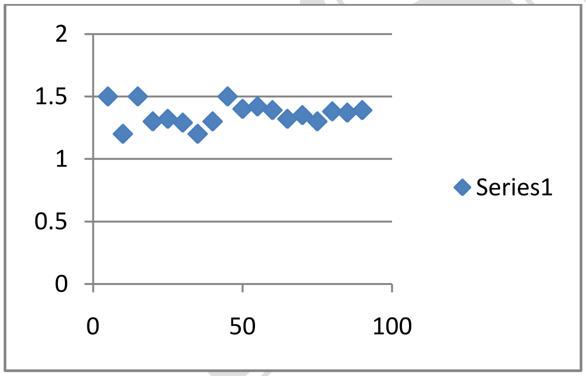
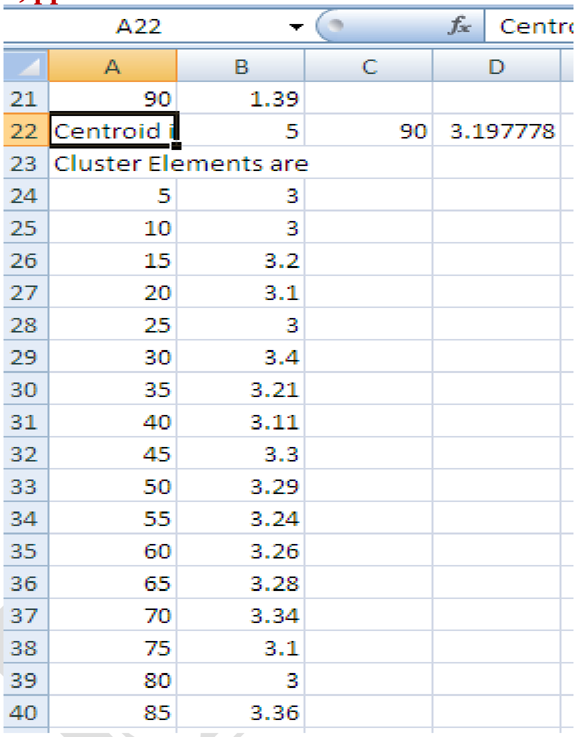


Fig. 4 Graph of Cluster I



	A	B	C	D
21	90	1.39		
22	Centroid i	5	90	3.197778
23	Cluster Elements are			
24	5	3		
25	10	3		
26	15	3.2		
27	20	3.1		
28	25	3		
29	30	3.4		
30	35	3.21		
31	40	3.11		
32	45	3.3		
33	50	3.29		
34	55	3.24		
35	60	3.26		
36	65	3.28		
37	70	3.34		
38	75	3.1		
39	80	3		
40	85	3.36		

Fig. 5 Output data of Cluster II

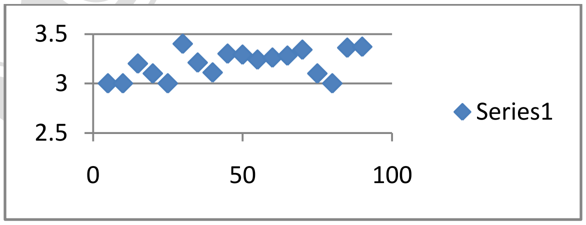


Fig. 6 Graph of Cluster II

From the snap shots of output data and its graphical representation it is clear that, we can observe the two lanes as the two clusters.

V Result

From all these experimental observations it is clear that after adding domain knowledge in the form of pair wise instance level constraints, we can have the elongated clusters. Elongated shape is the characteristic of the road lanes, hence after adding constraints in the K-Means algorithm we can make it intelligent to give the road lanes from the given GPS traces.

VI Conclusion

By adding domain knowledge in the k-means algorithm, we can have elongated clusters which are suitable for representing the road lanes. Thus the lane finding problem is solved by the constrained clustering algorithm. Again we can refine the clusters by using expectation maximization concept. Also we can modify the algorithm t automatically select the correct value of number of clusters, 'k'.

VII References

[1] Kiri Lou Wagstaff "Intelligent clustering with Instance – level constraints" Cornell University 2002

[2] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schroedl "Constrained K-means Clustering with Background Knowledge" Proceedings of the Eighteenth International Conference on Machine Learning, 2001

[3] Yihua Chen, John Krumm "Probabilistic Modeling of Traffic Lanes from GPS Traces", Microsoft Research, 2010.

[4] Dan Klein**,** Sepandar D. Kamvar, Christopher D. Manning "From Instance – level constraints to space – level constraints: Making the most of prior knowledge in data clustering". Department of Computer Science, Stanford University, Stanford, CA 94305-9040 USA

[5] Hui Yang, Jamie Callan "Near – duplicate detection by instance – level constrained clustering". Language Technologies Institute School of Computer Science Carnegie Mellon University

[6] Seth Rogers Pat Langley Christopher Wilson "Mining GPS data to augment road models". DaimlerChrysler Research and Technology Center

[7] Ian Davidson, Sugato Basu "A Survey of Clustering with Instance Level Constraints". ACM Transactions on Knowledge Discovery From Data Vol. w No. x z 2007.

[8] Kiri Wagsta, Claire Cardie "Clustering with Instance – level constraints" Proceedings Of Seventh International Conference on Machine Learning 2000.

[9] Duy Vu, Prem Melville, Mikhail Bilenko, Mayta Saar-Tsechansky "Intelligent Information Acquisition For Improved Clustering."

[10] Jung-Eun Park, Kyung-Whan Oh "Multi – Agent System For Intelligent Clustering" World Academy of Science, Engineering and Technology 11 2005)

[11] Data Clustering And Its Application (Raza Ali, Usman Gani, Aasim Saeed)