C-NBC: Neighborhood-Based Clustering with Constraints

Piotr Lasek

Chair of Computer Science, University of Rzeszów
ul. Prof. St. Pigonia 1, 35-310 Rzeszów, Poland
lasek@ur.edu.pl

Abstract. Clustering is one of most important methods of data mining. It is used to identify unknown yet interesting and useful patterns or trends in datasets. There are different types of clustering algorithms such as partitioning, hierarchical, grid and density-based. In general, clustering methods are considered unsupervised, however, in recent years the new branch of clustering algorithms has emerged, namely constrained clustering algorithms. By means of so-called constraints, it is possible to incorporate background knowledge into clustering algorithms which usually leads to better performance and accuracy of clustering results. Through the last years, a number of clustering algorithms employing different types of constraints have been proposed and most of them extend existing partitioning and hierarchical approaches. Among density-based methods using constraints algorithms such as *C-DBSCAN*, *DBCCOM*, *DBCluC* were proposed. In this paper we offer a new *C-NBC* algorithm which combines known neighborhood-based algorithm (*NBC*) and instance-level constraints.

Keywords: data mining, clustering, semi-supervised learning, constraints

1 Introduction

Clustering is one of well-known and often used data mining methods. Its goal is to assign data objects (or points) to different clusters so that objects that were assigned to the same clusters are more similar to each other than to objects assigned to another clusters [1]. Clustering algorithms can operate on different types of data sources such as databases, graphs, text, multimedia, or on any dataset containing objects that could be described by a set of features or relationships [2]. Since clustering algorithms do not require any external knowledge to be run (e.g. except parameters like k in the *k-Means* algorithm), the process of clustering, in opposite to classification, is also often referred to as an unsupervised learning. However, there has always been a natural need to incorporate already collected knowledge into algorithms to make them better both in terms of efficiency and quality of results. This need leads to the construction of a new branch of clustering algorithms based on so-called constraints.

Constraint-based clustering algorithms employ the fact, that in many applications, the domain knowledge in the form of labeled objects is already known or could be easily specified by domain experts. Moreover, in some cases such knowledge can be

automatically detected. Initially, researchers focused on algorithms that incorporated pairwise constraints on cluster membership or learned distance metrics. Subsequent research was related to algorithms that used many different kinds of domain knowledge [3].

The major contribution of this work is the offering of a constrained neighborhood-based clustering algorithm called *C-NBC* which combines the known *NBC* algorithm [5] and instance-level constraints such as *must-link* and *cannot-link*.

This paper is divided into five sections. In Section 1 we have given a brief introduction to clustering with constraints. Next, in Section 2, we shortly describe most known types of constraints and focus on instance-level constraints such as *must-link* and *cannot-link* constraints. In Section 3, the *Neighborhood-Based Clustering* (*NBC*) is reminded. Further, in Section 4 we present the proposed *C-NBC* algorithm. Conclusions are drawn and further research is briefly discussed in Section 5.

2 Instance-Level Constraints

In constrained clustering background knowledge can be incorporated into algorithms by means of different types of constraints. Through the years, different methods of using constraints in clustering algorithms have been developed [3]. Constraint-based methods proposed so far employ techniques such as modifying the clustering objective function including a penalty for satisfying specified constraints [7], clustering using conditional distributions in an auxiliary space, enforcing all constraints to be satisfied during clustering process [8] or determining clusters and constraints based on neighborhoods derived from already available labeled examples [9]. On the other hand, in the distance based methods, the distance measure is designed so that it satisfies given constraints [10, 11]. A few density-based constrained algorithms such as *C-DBSCAN* [4], *DBCCOM* [16] or *DBCluC* [17], have also been proposed.

Table 1. The notations related to instance-level constraints used in this work

Notation	Description
$C_{=}$	The set of pairs of points that are in a *must-link* relation.
$c_{=}(p_0, p_1)$	Two points p_0 and p_1 are in a *must-link* relation (must be assigned to the same resulting cluster).
$C_{=}(p)$	The set of points which are in a *must-link* relation with point p.
$C_{\neq}$	The set of pairs of points that are in a *cannot-link* relation.
$c_{\neq}(r_0, r_1)$	Two points r_0 and r_1 are in a *cannot-link* relation (must not be assigned to the same resulting cluster).
$C_{\neq}(r)$	The set of points which are in a *cannot-link* relation with point r.

Several types of constraints are known, but the hard instance-level constraints seem to be most useful since the incorporation of just few constraints of this type can increase clustering accuracy as well as decrease runtime. In [6] authors introduced two kinds of instance-level constraints: the *must-link* and *cannot-link* constraints. These constraints are simple, however they have interesting properties. For example *must-*

link constraints are symmetrical, reflexive and transitive, similarly to an equivalence relation. Generally, if two points, say p_0 and p_1, are in a *must-link* relationship (or, in other words, are connected by a *must-link* constraint), then these points, in a process of clustering, must be assigned to the same cluster c. On the other hand, if two points, say r_0 and r_1, are in a *cannot-link* relationship (or are connected by a *cannot-link* constraint), then these points must not be assigned to the same cluster c.

In the next part of the paper, when referring to *must-link* and *cannot-link* constraints, we will use the notations presented in Table 1.

3 Neighborhood-Based Clustering

The *Neighborhood-Based Clustering* (*NBC*) algorithm [5] belongs to the group of density based clustering algorithms. The characteristic feature of *NBC* is the ability to measure relative local densities. Hence, it is capable of discovering clusters of different local densities and of arbitrary shape. Below we remind the key definitions related to the *NBC* algorithm which will be used in the sequel.

Definition 1. (*ε-neighborhood*, or briefly $\varepsilon NN(p)$) *ε-neighborhood* of point p ($\varepsilon NN(p)$) is the set of all points q in dataset D that are distant from p by no more than ε; that is, $\varepsilon NN(p) = \{q \in D | dist(p,q) \leq \varepsilon\}$, where $dist$ is a distance function.

Definition 2. (k^+*-neighborhood*, or $k^+NN(p)$ in brief) k^+-neighborhood of point p ($kNN(p)$) is a set of k ($k > 0$) points satisfying the following conditions: $|kNN(p)| = k$ (∗), $\bigvee_{o' \in D \setminus kNN(p)} \bigvee_{o \in kNN(p)} dist(p,o') \geq dist(p,o)$ (∗∗).

Definition 3. (*punctured* k^+*-neighborhood, or briefly* $k^+NN(p^-)$) k^+-neighborhood of point p ($k^+NN(p^-)$) is equal to $\varepsilon' NN(p) \setminus \{p\}$, where $\varepsilon' = \max(\{dist(p,v) | v \in kNN(p)\})$.

Definition 4. (*reversed punctured* k^+*-neighborhood of a point p*, or $Rk^+NN(p)$ in brief) Reversed punctured k^+-neighborhood *of a point p* ($Rk^+NN(p)$) is the set of all points $q \neq p$ in dataset D such that $p \in k^+NN(q^-)$; that is: $Rk^+NN(p) = \{q \in D | p \in k^+NN(q^-)\}$.

Definition 5. (*neighborhood-based density factor of a point*) Neighborhood-based density factor of a point p ($NDF(p)$) is defined as $NDF(p) = |Rk^+NN(p)| / |k^+NN(p^-)|$. (Points having the value of the *NDF* factor equal to or greater than 1, are considered as dense.)

In order to find clusters, *NBC* starts with calculating values of NDF factors for each point p_i in a database D, $i = 0,1,\ldots,|D|$. Next, for each p_i, a value of NDF is checked. If it is greater than or equal to 1, then p_i is assigned to the currently created cluster c (identified by the value of $ClusterId$). Next, the temporary variable $DPSet$ for storing references to points, is cleared and each point, say q, belonging to $k^+NN(p_i^-)$ is assigned to c. If $q.ndf$ is greater than or equal to 1, then q is also added to $DPSet$. Otherwise, q is omitted and a next point from $k^+NN(p_i^-)$ is analyzed. Further, for each point from $DPSet$, say r, $k^+NN(r^-)$ is computed. All unclassified

points belonging to $k^+NN(r^-)$ are assigned to c and points having values of NDF greater than or equal to 1 are added to $DPSet$. Next, r is removed from $DPSet$. When $DPSet$ is empty, $ClusterId$ is incremented and a next point from D, namely p_{i+1}, is analyzed. Finally, if there are no more points in D having values of NDF factor greater than or equal to 1, then all unclassified points in D are marked as NOISE.

4 Clustering with Constraints

In this section we offer our new neighborhood-based constrained clustering algorithm called *C-NBC*. The algorithm is based on the *NBC* algorithm [5] but uses both *must-link* and *cannot-link* constraints for incorporating knowledge into the algorithm.

The *C-NBC* algorithm employs the same definitions as *NBC* which are used in a process of clustering to assign points to appropriate clusters or mark them as noise. In *NBC* three kinds of points can be distinguished: unclassified, classified and noise points. In our proposal, we introduced another kind of points, namely deferred points (Definition 6).

Definition 6. (*deferred point*) A point p is called deferred if it is involved in a *cannot-link* relationship with any other point or it belongs to a k^+-punctured neighborhood $k^+NN(q^-)$, where q is any point involved in a *cannot-link* relationship.

C-NBC (Figure 1) can be divided into two main steps. In the first step the algorithms works very similarly to *NBC*. It calculates *NDF* factors and performs clustering. The difference between *C-NBC* and *NBC* is that the former additionally determines deferred points and has to deal with *must-link* constraints when building clusters. In spite of the fact that in the first step the deferred points are not assigned to any cluster, these points are normally used to calculate *NDF* factors. In the second step *C-NBC* works so that it allocates deferred points to appropriate clusters. This is done by means of the *AssignDeferredPointsToClusters* function (Figure 2) which assigns points to clusters after checking if such an assignment if feasible. The detailed description of *C-NBC* is provided beneath. The key variables and notations related to *C-NBC* are explained in Table 2.

Table 2. The auxiliary variables and notations used in pseudo-code of *C-NBC* if Figure 1

Variable or notation	Description
$ClusterId$	The auxiliary integer variable used for storing currently-created cluster's identifier.
$p.ClusterId$	By using such a notation we refer to a $ClusterId$ related to point p.
$p.ndf$	Such a notation is used to refer to a value of the *NDF* factor associated with point p.
R_d, R_t	The auxiliary variables for storing deferred points.
$DPSet$	The variable for storing dense points. It is used for in an iterative process of assigning points to clusters.

Algorithm *C-NBC*(D, k, $C_=$, $C_{\neq}$)

Input: D the input not clustered dataset
k the parameter of the *C-NBC*
$C_=$, $C_{\neq}$ the sets of pairs of points connected by *must-link* or *cannot-link* constraints

Output: The clustered set with clusters satisfying *cannot-link* and *must-link* constraints.

```
R_d ← ∅;   ClusterId = 0;  // initialization of a set of deferred points R_d and ClusterId
label all points in D as UNCLASSIFIED;  // p.ClusterId = UNCLASSIFIED
CalcNDF(D, k);  // calculate values of NDF factor for every point in D
for each point q involved in any constraint from C_= or C_≠ do
    label q and points in k⁺NN(q⁻) as DEFERRED  // label points as DEFERRED
    add q to R_d;  // add DEFERRED points to R_d
endfor
ClusterId = 0;
for each unclassified point p in D such that p.ndf ≥ 1 do
    p.ClusterId = ClusterId;
    clear DPSet;
    for each point q ∈ k⁺NN(p⁻)\R_d do
        q.ClusterId = ClusterId;  // set point's cluster identifier
        if (q.ndf ≥ 1) then add q to DPset; endif
        add all points r from C_=(q) such that r.ndf ≥ 1 to DPset;
    endfor
    while (DPSet ≠ ∅) do          // expanding a currently created cluster
        s = first point in DPset;
        for each unclassified point t in k⁺NN(s⁻)\R_d do
            t.ClusterId = ClusterId;  // set point's cluster identifier
            if (t.ndf ≥ 1) then add t to DPset; endif
            add all points u from C_=(t) such that u.ndf ≥ 1 to DPset;
        endfor
        remove s from DPset;
    endwhile
    ClusterId++;
endfor
label unclassified points in D as NOISE;
AssignDeferredPointsToClusters(D, R_d, k, C_≠);
```

Fig. 1. The pseudo-code of the *C-NBC* algorithm. Most important differences between *NBC* and *C-NBC* are marked with gray color

The *C-NBC* algorithm starts with the *CalcNDF* function. After calculating the *NDF* factors for each point from *D*, the deferred points are determined by scanning pairs of *cannot-link* connected points. Such a points are added to an auxiliary set R_d.

Then, the clustering process is performed in the following way: for each point p which was not marked as DEFERRED, it is checked if $p.ndf$ is less than 1. If $p.ndf < 1$, then p is omitted and a next from $DPSet$ is checked. If $p.ndf \geq 1$, then p as a dense point is assigned to the currently-created cluster identified by the current value of $ClusterId$.

Next, the temporary variable $DPSet$ is cleared and each not deferred point, say q, belonging to $k^+NN(p^-)\backslash R_d$ is assigned to the currently-created cluster identified by the current value of the $ClusterId$ variable. Additionally, if $q.ndf \geq 1$, then it is assigned to $DPSet$ as well as all dense points which are in a *must-link* relation with q.

Function *AssignDeferredPointsToClusters*(D, R_d, k, $C_{\neq}$)

Input: D the input not clustered dataset
R_d the set of points marked as deferred
k the parameter of the *C-NBC* algorithm
$C_{\neq}$ the set of *cannot-link* constraints

Output: The clustered set with clusters satisfying *cannot-link* and *must-link* constraints

```
R_t ← R_d;        // saving contents of R_d in a temporary set R_t
do begin
    R_a ← ∅; // a temporary set for storing deferred points assigned to any cluster
    foreach point p in R_t do begin
        foreach point q in k⁺NN(p⁻) do
            if (q.ndf ≥ 1 and q.ClusterId > 0 and q ∉ R_d)
                if (CanBeAssigned(p, q.ClusterId)) and // checking if p can be assigned to
                                                       // cluster identified by q.clusterId
                    (CanBeAssigned(p.nearestCannotLinkPoint, q.ClusterId)) then
                    p.ClusterId = q.ClusterId; add p to R_a;
                    break;
                endif
            endif
        endfor
        remove R_a from R_t;
    endfor
while (R_a ≠ ∅)
```

Fig. 2. The pseudo-code of the *AssignDeferredPointsToClusters* function

Next, for each unclassified point from $DPSet$, say s, its punctured k^+-neighborhood is determined. Each point, say t, which belongs to this neighborhood and has not been labeled as deferred yet is assigned to the currently created cluster and if its value of *NDF* is equal to or greater than 1, is added to $DPSet$. Moreover, all dense points which are in a *must-link* relation with t are added to $DPSet$ as well. Later, s is removed from $DPSet$ and next point from $DPSet$ is processed. When $DPSet$ is emptied, then $ClusterId$ is incremented.

After all points from D are processed, unclassified points are marked as noise by setting the values of their $ClusterId$ attribute to *NOISE*. However, in order to process the deferred points, the *AssignDeferredPointsToCluster* function is invoked. The function performs so that for each deferred point p it finds the nearest point q assigned to any cluster and checks whether it is possible (in accordance with *cannot-link* constraints) to assign p to the same cluster as q. Additionally, the function checks if the assignment of p to a specific cluster will not violate previous assignments of deferred points.

In order to test the efficiency of the proposed *C-NBC* algorithm we performed a number of basic experiments using the following datasets: a manually created dataset containing 2800 data points, the known four dimensional iris dataset [13] and the birch dataset [14]. The main goal of the experiments was to test the influence of a number of constraints used in the process of clustering on the efficiency of the proposed algorithm. In Table 2 we compare the run-times of the *NBC* and *C-NBC* algorithms. Both implementations of the algorithms employ the same index structure,

namely the *R-Tree* [15]. When performing experiments using *C-NBC* we were changing the number of *must-link* and *cannot-link* constraints from 2 to 250 except for the Iris dataset, which was not numerous enough to use number of both types of constraints greater than 50. Since additional operations must be performed, it was obvious for us that the number of constraints would have a negative impact on the efficiency of the algorithm. However, we did not encounter any crucial increase (at most about 2 times for the largest dataset and 250 *must-link* and 250 *cannot-link* constraints) in the run-time of clustering using the *C-NBC* algorithm. For this reason, similarly to *NBC*, *C-NBC* can be considered as effective, efficient, and easy to use.

Table 3. The results of the experiments. Run-times are given in milliseconds. The value of k (the parameter of *C-NBC* algorithm) is equal to 15. Number of constraints concerns both *must-link* and *cannot-link* constraints. $|D|$ – number of points in a dataset, d – number of dimensions, * - a number of constraints was greater than the cardinality of a dataset

Algorithm	**Number of constraints**	**Run-times**		
		Manually created $\|D\| = 2800$, $d = 2$	Iris $\|D\| = 250$, $d = 4$	Birch $\|D\| = 10000$, $d = 2$
NBC	0	3284	67	13385
C-NBC	2	3500	101	14132
	10	4172	105	14969
	50	6067	*	21360
	250	6457	*	29931

5 Conclusions and further research

In recent years a number of algorithms employing different kinds of constraints have been proposed. Most commonly used constraints are instance constraints which are widely employed in enriching clustering algorithms to utilize additional background knowledge. However, among density based clustering algorithms, so far only the *DBSCAN* algorithm [12] was adapted to use instance constraints.

In this paper we have offered *C-NBC*, the density based algorithm for clustering under instance constraints which was based on the known *NBC* algorithm. Our preliminary experiments, confirm that the algorithm performs in line with the assumptions. In other words, it works so that *must-link* connected points are assigned to the same clusters and *cannot-link* connected points are assigned to different clusters. Moreover, for datasets tested, it turns out that constraints have a little effect on the efficiency of the algorithm. In spite of this, in our future research we would like to still focus on performance of constrained based clustering and on proposing other methods of ensuring constraints during clustering process for larger and high dimensional datasets.

References

1. J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.
2. S. Basu, I. Davidson, and K. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications, 1 edition*, vol. 45. 2008, pp. 961–970.
3. I. Davidson and S. Basu, "A Survey of Clustering with Instance Level Constraints," *ACM T. Knowl. Disc. Data*, vol. w, pp. 1–41, 2007.
4. C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "C-DBSCAN: Density-Based Clustering with Constraints," in *RSFDGr'07: Proc. of the International Conf. on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing held in JRS07*, 2007, vol. 4481, pp. 216–223.
5. S. Zhou, Y. Zhao, J. Guan, and J. Huang, "A Neighborhood-Based Clustering Algorithm," in *Advances in Knowledge Discovery and Data Mining SE - 43*, vol. 3518, T. Ho, D. Cheung, and H. Liu, Eds. Springer Berlin Heidelberg, 2005, pp. 361–371.
6. K. Wagstaff and C. Cardie, "Clustering with Instance-level Constraints," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 1103–1110.
7. I. Davidson and S. Ravi, "Clustering With Constraints: Feasibility Issues and the k-Means Algorithm," in *Proceedings of the 2005 SIAM Int. Conf. on Data Mining*, pp. 138–149.
8. K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained K-means Clustering with Background Knowledge," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 577–584.
9. S. Basu, A. Banerjee, and R. J. Mooney, "Semi-supervised Clustering by Seeding," in *Proceedings of the 19th International Conference on Machine Learning*, 2002, pp. 27–34.
10. T. H. Tomboy, A. Bar-Hillel, and D. Weinshall, "Boosting Margin Based Distance Functions for Clustering," in *In Proceedings of the Twenty-First International Conference on Machine Learning*, 2004, pp. 393–400.
11. H. Chang and D.-Y. Yeung, "Locally Linear Metric Adaptation for Semi-supervised Clustering," in *Proc. of the 21st International Conf. on Machine Learning*, 2004, pp. 153–160.
12. M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.
13. C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases," *University of California*. p. http://archive.ics.uci.edu/ml/, 1998.
14. T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH : A New Data Clustering Algorithm and Its Applications," *Data Min. Knowl. Discov.*, vol. 1, pp. 103–144, 1997.
15. A. Guttman, "R Trees: A Dynamic Index Structure For Spatial Searching," in *ACM SIGMOD Int. Conf. on Management of Data - SIGMOD '84*, 1984, p. 47–53.
16. Duhan, N., & Sharma, A. K. (2011). DBCCOM: Density Based Clustering with Constraints and Obstacle Modeling. In *Contemporary Computing SE - 24* (Vol. 168, pp. 212–228). Springer Berlin Heidelberg.
17. Zaiane, O. R., & Lee, C.-H. L. C.-H. (2002). Clustering spatial data when facing physical constraints. *2002 IEEE International Conference on Data Mining, 2002. Proceedings*.