Semi-Supervised Density-Based Clustering

Levi Lelis
Department of Computing Science
University of Alberta, Edmonton, Canada
santanad@cs.ualberta.ca

Jörg Sander
Department of Computing Science
University of Alberta, Edmonton, Canada
joerg@cs.ualberta.ca

Abstract

Most of the effort in the semi-supervised clustering literature was devoted to variations of the K-means algorithm. In this paper we show how background knowledge can be used to bias a partitional density-based clustering algorithm. Our work describes how labeled objects can be used to help the algorithm detecting suitable density parameters for the algorithm to extract density-based clusters in specific parts of the feature space. Considering the set of constraints established by the labeled dataset we show that our algorithm, called SSDBSCAN, automatically finds density parameters for each natural cluster in a dataset. Four of the most interesting characteristics of SSDBSCAN are that (1) it only requires a single, robust input parameter, (2) it does not need any user intervention, (3) it automatically finds the noise objects according to the density of the natural clusters and (4) it is able to find the natural cluster structure even when the density among clusters vary widely. The algorithm presented in this paper is evaluated with artificial and real-world datasets, demonstrating better results when compared to other unsupervised and semi-supervised density-based approaches.

1 Introduction

Semi-supervised learning has bicdm09xeen studied recently as a method for improving generalization in Machine Learning models [10]. The general idea is to augment the information given by labeled training samples with useful information from unlabeled samples. The idea behind semi-supervision is not limited to using unlabeled samples to help algorithms that typically use labeled data. We can also use labeled samples to help algorithms that typically use unlabeled data. Clustering, also called unsupervised learning, tries to group unlabeled objects of a dataset based on a measure of similarity. Labeled objects could be used in clustering algorithms to help determine the groups. Clustering algorithms that make use of class membership information for some of the samples are called *semi-supervised clustering algorithms*.

Clustering algorithms try to extract non-obvious information about natural groups in the data from a collection of unlabeled objects. Sometimes, however, the user might have more than just a collection of unlabeled objects. The user might have domain-knowledge about class membership for some of those objects. By exploiting this type of information the user can bias the process of clustering in order to get more meaningful results. Our work describes how labeled objects can be used to help a density-based clustering algorithm by detecting suitable density parameters to extract density-based clusters in specific parts of the feature space.

The notion of density-based clustering that we base our algorithm on is similar to the notion of density-based clustering introduced by Ester *et al.* [7] for the DBSCAN algorithm. DBSCAN requires two parameters, a minimum number of points $MinPts \in \mathbb{N}$, and a radius $\epsilon \in \mathbb{R}$, which define the density level at which clusters will be detected. These parameters are notoriously difficult to determine in real world data sets, which limits the applicability of density-based algorithms in an automatic KDD process. In some cases, where the density among clusters differ widely, there is not even a single set of parameter values for ϵ and $MinPts$ that allows to extract the real cluster structure of a dataset for DBSCAN [8]. Figure 1 show an example where no global density threshold exists that can separate all three natural clusters, and consequently, DBSCAN cannot find the intrinsic cluster structure of the dataset. The figures depict the resulting clusters found by DBSCAN for two different values for ϵ and a fixed value for $MinPts$; noise objects in these figures are shown as circles. Figure 1(left) shows that DBSCAN can separate the two denser clusters, A and B, with a suitable parameter value for ϵ, but then, it has to mark the third cluster, C, as noise. By changing the value of ϵ in order to detect clusters of lower density, DBSCAN merges the two denser clusters and separates it from the third cluster (denoted by the letter B in figure 1(right)).

DBSCAN's authors stated that, "...ideally, we would have to know the appropriate parameters ϵ and $MinPts$

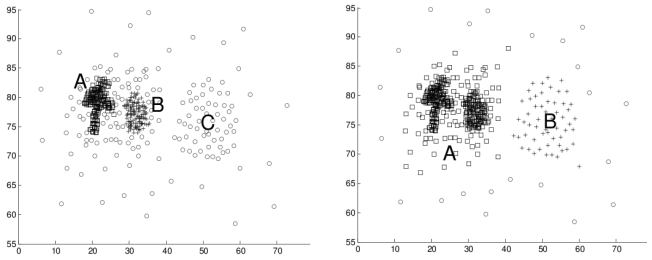

Figure 1. In this case, no global set of parameters extracts the real structure

of each cluster and at least one point from the respective cluster. Then, we could retrieve all points that are density-reachable from the given point using the correct parameters. But there is no easy way to get this information in advance for all clusters in the database" [7]. In our case, we assume that information about cluster membership for a small subset of objects is available, which could provide valuable information about the density parameters that characterize different clusters. Using available background knowledge in this sense is exactly what is accomplished by the Semi-Supervised DBSCAN (SSDBSCAN): given a set of labeled objects, and assuming that the labels are consistent with the density-based clustering structure of the data, SSDBSCAN finds values for ϵ given a fixed value of $MinPts$ that extract the natural clusters of a dataset, while separating objects with different labels. We show with an artificial dataset that SSDBSCAN is able to extract meaningful clusters even when the density between clusters vary widely. SSDBSCAN was also evaluated with real-world benchmark datasets and compared with another semi-supervised density-based clustering algorithm.

The rest of the paper is organized as follows. The next section briefly surveys other semi-supervised clustering algorithms. In section 3 we formally specify our assumptions, define the problem of finding density parameters values, and present an efficient algorithm that solves the problem. Section 4 shows and discusses the empirical results. Finally, section 5 concludes the paper.

2 Related Work

Wagstaff *et al.* [9] defined the concept of two basic kinds of pairwise constraints that made the insertion of domain knowledge into the clustering(K-means in this case) process possible: the *must-link* and *cannot-link* constraints. A must-link constraint between two objects indicates that those two objects should be in the same cluster, while a cannot-link constraint indicates the opposite, that two objects should not be in the same cluster. Wagstaff *et al.* showed with benchmark datasets that it is possible to bias the process of clustering while respecting the constraints, getting more meaningful clusters as a result. A good number of papers followed Wagstaff's ideas, modifying and improving on the usage of constraints for the K-means algorithm; [3] [6] to cite a few.

Böhm and Plant [4] presented HISSCLU, a hierarchical density-based clustering algorithm based on OPTICS [1]. HISSCLU can be described in two stages. In the first stage, given a set of labeled objects, HISSCLU starts the OPTICS expansion simultaneously from all the labeled objects and generates as many reachability-plots as the number of labeled objects, each one representing a cluster; during the label expansion they use a method to change the distance between points that resembles the distance learning [3]. The reachability-plots are reordered and concatenated with each other, producing one single plot. In the second stage, a cut at level ϵ is made in the plot to extract the clusters [1].

It is important to notice that HISSCLU is not able to extract the natural cluster structure from a dataset if the plot generated in the first stage of the algorithm represents a distribution where the density varies widely between clusters, as it also uses only one single cut. As for DBSCAN, defining the value of a single cut corresponding to a single density level is difficult and requires the user often to perform a trial and error process, which makes the algorithm unsuitable for an automatic KDD process.

3 Semi-Supervised Density-Based Clustering

The basic idea of density-based clustering is that clusters are dense regions in the feature space separated by regions of lower density. Ester *et al.*[7] formalized this idea by defining density-based clusters as a set of spatially connected points (including their neighborhood) whose density estimate exceeds a certain threshold. Density-based clusters, spatial connectedness, and density estimate are specified using two parameters, $\epsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$ as following, assuming a dataset D:

- *Core Objects*. An object $p \in D$ is called core-object w.r.t ϵ and $MinPts$ if $|N_\epsilon(p)| \geq MinPts$, where $|N_\epsilon(p)|$ is the ϵ-Neighborhood of a point p.

- *Density-Reachable*. An object $q \in D$ is density-reachable from an object $p \in D$ (directly or transitively) w.r.t ϵ and $MinPts$ if there is chain of objects $p_1, ..., p_n$ in D, $p_1 = p$, $p_n = q$ such that $p_{i+1} \in N_\epsilon(p_i)$, and p_i is a core object.

- *Density-Connected*. An object $p \in D$ is density-connected to $q \in D$ w.r.t ϵ and $MinPts$ if there is an object $v \in D$ that both p and q are density-reachable from.

- *Density-Based Cluster*. A density-based cluster C w.r.t ϵ and $MinPts$ is a non-empty subset of D, satisfying:

  1. $\forall p, q \in D$: if $p \in C$ and q is density-reachable from p w.r.t ϵ and $MinPts$ then $q \in C$.
  2. $\forall p, q \in C$: p is density-connected to q w.r.t ϵ and $MinPts$.

Ester *et al.* proposed an algorithm, called DBSCAN, that, given a dataset D, returns the density-based clusters (and a set of noise points, i.e., point not in clusters) of D based on the above definition. DBSCAN requires as input global values for ϵ and $MinPts$, which are typically difficult to set, and in many cases, a global density level will not reveal the complete cluster structure in the data.

The problem of finding global density parameters has also been observed by Ankerst *et al.* [1] who propose a hierarchical version of DBSCAN called OPTICS. For OPTICS, $MinPts$ is set to a fixed value so that density-based clusters of different densities are characterized by different values for ϵ. The problem with OPTICS is that it computes only a hierarchical representation of the clustering structure, which has to be analyzed interactively and from which clusters are typically determined manually based on a graphical representation called the reachability plot.

Our goal is to find density-based clusters automatically, especially in those cases where no global density-threshold exists, in cases when a user has some knowledge about the clusters in a data set, in form of a small subset of objects in D for which class labels are known. For this purpose, we will also fix the value of $MinPts$ for all clusters, as in Ankerst *et al.*[1] (making $MinPts$ the only parameter of our method), and try to find the ϵ values for clusters of different densities.

Suppose we have a dataset D and a supervised subset of D, $D_L = \{o_i\}_{i=1}^n$, of n labeled objects o_i, where $L(o_i)$ denotes the label of o_i, and there is at least one (possibly more) labeled objects from each class in D_L. In order to utilize this information for density-based clustering, we have to assume that the class labels are consistent with the clustering structure of the data set in the following sense:

Definition 1. ***(label consistency)*** *Let $o_i, o_j \in D_L$ such that $L(o_i) \neq L(o_j)$, i.e., the labels of o_i and o_j are different. Then, o_i and o_j are* label consistent *with each other (w.r.t. a value of $MinPts$) if there are two density-based clusters C_i and C_j (w.r.t. ϵ_1 and ϵ_2, respectively) such that $o_i \in C_i$ and $o_j \in C_j$, and C_i and C_j are density separable, i.e., $C_i \cap C_j = \emptyset$*[1].

Label consistency only requires different labels to belong to different clusters; under this assumption, a single class can still have multiple modes, in other words, objects in different clusters can have the same label, only in a single cluster, the labels have to be the same.

The problem we want to solve, using the information about labeled data for density-based clustering, is to find suitable ϵ values for each labeled object in D_L, i.e., find $\{\epsilon_i\}_{i=1}^n$, where n is the number of labeled objects, so that each object in D_L is contained in a density-based cluster and all pairs of objects in D_L with different labels are label consistent. For a given data set, there may be several sets of values $\{\epsilon_i\}_{i=1}^n$ that satisfy these conditions. We are interested in the set that results in the largest possible clusters that satisfy the cluster assumption in order to assign labels to the largest possible number of unlabeled objects based on the labeled object(s).

3.1 The algorithm

To derive an algorithm for our problem statement, we first observe that two objects p and q will be in the same cluster w.r.t. $MinPts$ and some value for ϵ, if they are density-connected to each other w.r.t. $MinPts$ and ϵ. Since density-connectivity between p and q requires a chain of objects $o_1, \ldots, o_k$ between p and q where $dist(o_i, o_{i+1}) \leq \epsilon$, there is a smallest ϵ, say ϵ' for which such a chain can exist, i.e., for which p and q are density-connected.

To separate p and q, in order to achieve label consistency in case they have different labels, we can choose values strictly smaller than ϵ' as a density threshold. However, we are interested in the largest possible separate clusters C_1, C_2 containing p and q, respectively. For a given data set, the set of density thresholds for which the resulting clusters containing p and q are maximal (i.e., equal to C_1 and C_2), has a smallest element.

Lemma 1. *Assume $p, q \in D_L$ with different labels, and let ϵ' be the smallest ϵ value for which p and q are density-connected w.r.t. $MinPts$ and ϵ. Furthermore, let C_1 and C_2 be the largest, separate density-based clusters w.r.t. $MinPts$ and a "cut value" $\leq \epsilon'$ that exist such that $p \in C_1$, $q \in C_2$. The smallest such "cut value" is equal to the length of the largest edge in the set of all density-connection paths between objects within C_1, respectively within C_2.*

Proof. (Sketch) The cut value cannot be smaller than the length of the largest edge in the set of all density-connection paths between objects in C_1, respectively C_2, otherwise at least one point connected in one of the two clusters with that edge length will no longer be included in that cluster, which no longer would be maximal. □

Definition 2. ***(separation density threshold)*** *Assume $p, q \in D_L$ with different labels, and let ϵ' be the smallest ϵ value for which p and q are density-connected w.r.t. $MinPts$ and*

[1] Note that two density-based clusters w.r.t. the same value of $MinPts$ and two different values ϵ_1, respectively ϵ_2, are either disjunct or one is completely contained in the other one.

ϵ. The separation density threshold for p and q is the length of the largest edge in the set of all density-connection paths between objects within C_1, respectively within C_2.

Given the set of points in D_L that have to be separated in order to achieve label consistency, we have to choose a density threshold for each $p \in D_L$ so that it falls into a density-based cluster that is separated from all $q \in D_L$ that have a different label. For a fixed $p \in D_L$ this density threshold is given by the smallest separation density threshold for p and q, among all $q \in D_L$ with $L(p) \neq L(q)$. This density threshold will separate a maximal cluster containing p from the "closest" cluster containing a point with a different label (closest in terms of density reachability path distance). This value, because it is the smallest of $p's$ separation density thresholds, will also separate p from any other point in D_L with a different label. This suggests an efficient algorithm that extracts one cluster at a time, in a similar fashion as OPTICS. We can start from a single labeled object o and expand a cluster until an object with a different label becomes density reachable from o. For that purpose, we define the core distance of a point o as in [1] as the smallest radius r that makes o a core point w.r.t. $MinPts$, i.e., for which $|N_r(n)| = MinPts$, which is the $MinPts$-nearest-neighbor distance of o.

Definition 3. $\forall o \in D : cDist_{MinPts}(o) =$ *MinPts-nearest-neighbor distance of o.*

Using the core distance, we can define a distance, called $rDist$, between all pairs of objects p, q, which indicates the smallest value for ϵ at which q and q are core points and directly density connected[2]:

Definition 4. $\forall p, q \in D :$
$rDist(p, q) = max(cDist(p), cDist(q), dist(p, q))$

Using these notions, we can construct a density-based cluster C containing a labeled point p by adding first p to C and then iteratively adding the next closest point in terms of $rDist$ to C. This is essentially what OPTICS does when starting with the point p; OPTICS also keeps track of the core distance and the reachability distance of each point at the time it is added. Conceptually, this is the same as constructing a minimum spanning tree (MST) for a complete graph where the set of vertices equals D and the edge weights are given by $rDist$. The difference here is that once a point q is added that has a label different from the label of p, the algorithm "backtracks" to the point o with the largest $rDist$ before adding q; this value for $rDist$ corresponds to the smallest ϵ value at which p and q are still density-connected. The current expansion stops and includes all points up to but excluding o , having constructed a maximal cluster C containing p. The largest $rDist$ value in this set of point is the separation density threshold for p and q.[3] Then, the algorithm repeats to extract the next cluster for the next labeled object. The pseudo-code is given below.

For simplicity, we present our algorithm in the same way as Prim's algorithm for constructing a MST found in Cormen *et al.* [5]. The algorithm assumes conceptually a complete graph built upon the dataset D, $D_L \subseteq D$ is the labeled dataset, and $rDist$ determines the edge weights; the output is a file with the labeled objects. After presenting the algorithm we discuss some implementation details that improve the memory usage and runtime of the algorithm.

Algorithm 1 SSDBSCAN(D, D_L)

```
1:  for all p ∈ D_L
2:    for all q ∈ D
3:      key[q] ← ∞
4:    key[p] ← 0
5:    Q ← D
6:    while Q ≠ ∅
7:      q ← EXTRACT-MIN(Q)
8:      list.insert(q)
9:      if q ∈ D_L and q.label ≠ p.label
10:       WRITE − OBJECTS(p, list)
11:       break
12:     for all o ∈ Adj[q]
13:       if o ∈ Q and rDist(q, o) < key[o]
14:         key[o] ← rDist(q, o)
```

Algorithm 2 WRITE-OBJECTS($p, list$)

```
1: i = arg max_x list(x)
2: for o = 1 to i − 1
3:   list[o].label = p.label
4:   write(list[o])
5:   list.clean()
```

The algorithm starts building an MST according to Prim's algorithm from an arbitrary vertex $p \in D_L$ and the process stops when either all objects are added to the MST or when an object with a different label than the label of p is

[2]Our definition of $rDist$ is similar to the definition of the *reachability distance* in Ankerst *et al.* [1], but differs in that it is symmetric, which avoids the problems with the original definition of reachability distance is that it does not deal with "border points" correctly when defining a cut-level through the OPTICS plot; as Ankerst *et al.* put it "Only some border objects may be missed when extracted by the algorithm ExtractDBSCAN-Clustering" [1]. With our definition of $rDist$ only the core points of a cluster will be extracted for a given cut level; border points can easily and correctly be added in a separate step (if desired).

[3]The fact that all points before adding o belong to a density-based cluster w.r.t. $MinPts$ and ϵ equal to the separation density threshold follows from the way Prim's algorithm constructs an MST: in each step of the algorithm, the point that has the smallest edge weight to one of the points in the current MST is added next.

added to the tree. The case when an object q with a different label is added to the tree (lines 9-11) indicates that the MST already "crossed" the border between two clusters that should be separated, and the algorithm looks for the currently largest edge that connects p and q in the set of points that were added before q to the current cluster/MST. In line 8 of the algorithm we keep a list of all objects that were added to the MST; they are sorted by the order in which they are added to the list. This list is given to the procedure WRITE-OBJECTS together with the root of the MST ($list$ and p, respectively). By the time WRITE-OBJECT is called, two objects with different labels have been added and the object added with the edge of highest $rDist$ value is the smallest ϵ value for which the two objects are density connected. In line 1 of the procedure WRITE-OBJECT the index (denoted by i) of that object in the ordered list is returned (object is denoted by x), and all objects from the beginning of the list up to x, excluding x, are written to an output file, assigning them the same label as the root object p. After writing the objects to a file, the data structures are re-initialized and the whole process is repeated for another object in D_L. In the end, all objects that belong to a cluster are written to an output file, while noise objects are left out. Lines 12-14 of SSDBSCAN show the standard procedure in Prim's algorithm to update the edge values of adjacent vertices of an object q ($Adj[q]$). SSDBSCAN can be seen as a procedure that calls Prim's algorithm a number of times equal to the number of labeled objects. Since the labeled dataset is finite, the algorithm will eventually terminate.

For clarity of presentation, we have omitted an obvious improvement of the algorithm which removes the redundancy in case there are multiple objects in D_L that lead to the same cluster: if we are creating an MST from an object whose label is c and if another object q with the same label c is inserted into the tree, we can remove q from the labeled dataset D_L in the algorithm so that we do not construct the same cluster starting from q in a later iteration.

In the implementation of SSDBSCAN shown above the input is conceptually a complete graph representing a dataset D with N objects, which has a space complexity of $O(N^2)$. However, we do not need the complete graph in main memory at any point in time, since we can compute the weights for edges ($rDist$ for pairs of vertices) as needed, resulting in an effective space complexity of $O(N)$.

SSDBSCAN calls Prim's algorithm a number of times equals to the number of objects in the labeled dataset. Prim's algorithm runs in $O(E \lg V)$ [5], which in our case corresponds to $O(N^2 \lg N)$, giving SSDBSCAN, a final time-complexity of $O(nN^2 \lg N)$, where n is the number of labeled objects in the labeled dataset. This time complexity can be improved by changing the heap implementation used in Prim's algorithm. If we use a Fibonacci heap [5] to implement the priority queue, our running time improves to $O(nN^2 + nN \lg N)$, that is, $O(nN^2)$. In practice we expect n to be very small, i.e., $N \gg n$.

4 Empirical Results

In this section we present the empirical results of SSDBSCAN and compare it with DBSCAN and HISSCLU. We will first discuss the result of SSDBSCAN on the artificial dataset shown in figure 1. As discussed before, DBSCAN cannot find the real cluster structure in that example. SSDBSCAN (using $MinPts = 2$), however, finds the intrinsic density of each cluster when given three labeled objects, one from each cluster (figure omitted due to lack of space).

The experiment with the artificial dataset is important for understanding how the algorithm works in an idealized case. In real world problems some of the assumptions made for the algorithm might not hold, and experiments with benchmark datasets (UCI datasets [2]) are the best way of verifying the effectiveness of SSDBSCAN. We evaluate how well SSDBSCAN performs when compared with the best global ϵ value from a large set of possible values for DBSCAN in a given dataset. In order to find the best ϵ value we try all possible ϵ values from 0.1 up to the maximum possible direct reachability distance in the data set, in increments of 0.1 (e.g., if the highest direct density reachability value is 10, we will try all ϵ values: 0.1, 0.2, ..., 10). The value that results in the best performance is shown in the graphs for DBSCAN. For the performance measure we used the Rand Statistic [8], which measure the agreement between two sets of clusters X and Y for the same set of n objects as: $R = \frac{a+b}{\binom{n}{2}}$, where a is the number of pairs of objects assigned to the same cluster in both X and Y, and b is the number of pairs of objects assigned to different clusters in both X and Y.

We also compared SSDBSCAN with HISSCLU. However, the comparison with HISSCLU is not straightforward, since it requires user intervention to analyze its plots and decide where a cut should be made. In addition, HISSCLU has three other input parameters which heavily influence the algorithm's performance. Therefore, we compared HISSCLU and SSDBSCAN with the datasets used by Böhm and Plant [4], for which the authors provided values for the input parameters and for the cut level. We used $MinPts = 3$ in all experiments for DBSCAN and SSDBSCAN. Noise objects were assigned to the cluster of the closest core object, for all three algorithms, DBSCAN, HISSCLU and SSDBSCAN. It is also important to state that all the results presented in this section are computed on the unlabeled dataset, not including the objects for which labels were provided to the clustering algorithm. The following UCI datasets were used in our experiments (after the name of each dataset we state the number of instances, attributes and classes in that dataset): ecoli (336; 8; 8), glass (214; 9; 7), liver (345; 6;

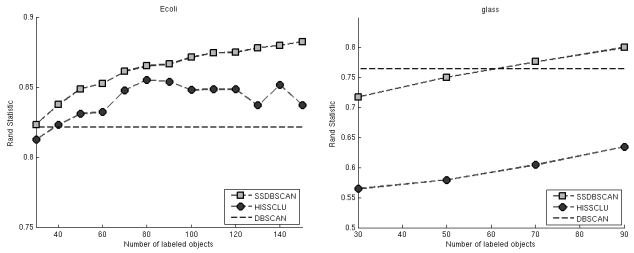

Figure 2. Results for the ecoli and glass

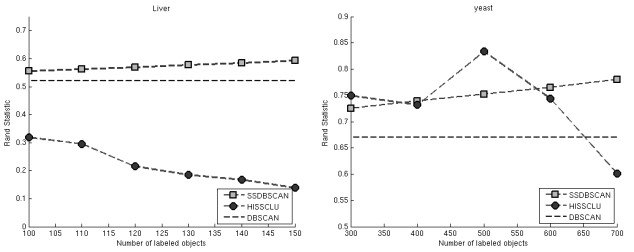

Figure 3. Results for the liver and yeast

2) and yeast (1484; 8; 10). The labeled objects were randomly selected, independently of the number of classes in the dataset. The following parameters were used for HISSCLU, as suggested in its paper: ecoli ($MinPts = 5, \rho = 2, \xi = 0.5, k = 0.2$), glass ($MinPts = 3, \rho = 200, \xi = 5, k = 0.2$), liver ($MinPts = 5, \rho = 20, \xi = 10, k = 0.9$), yeast ($MinPts = 5, \rho = 5, \xi = 5, k = 0.2$). The input parameters for the glass dataset were not specified in Böhm and Plant [4], therefore we set $MinPts = 3$, which is the same value used for SSDBSCAN; for ρ and ξ we tried different values and we chose the ones that yielded the best Rand Statistic on the whole dataset.

The results for SSDBSCAN are an average over 1000 trials, apart from yeast, for which 20 runs were averaged; the results for HISSCLU are an average over 100 trials for all except yeast, which was averaged over 10 runs; the results for DBSCAN are deterministic as it does not use the labels.

The results are depicted in figures 2 - 3. Each plot has a horizontal line that shows the best performance that DBSCAN can achieve and a curve depicting the performance of SSDBSCAN and HISSCLU when using a number of randomly selected labeled objects. As can be observed, SSDBSCAN clearly outperforms the best cut value of DBSCAN, showing that our approach is indeed taking advantage of the background knowledge provided by the labeled dataset. The results also show in most cases a significantly better performance of SSDBSCAN compared to HISSCLU.

In some of the experiments, the performance of HISSCLU dropped when adding more labeled objects. Such a drop is rare and less pronounced for SSDBSCAN. There is a discussion in the literature regarding of the usefulness of a set of constraints [6]. It is observed that some constraints might actually decrease the performance of an algorithm, and according to our experiments, SSDBSCAN turns out to be more robust in this respect than HISSCLU.

5 Conclusion

In this work we presented a novel semi-supervised density-based clustering algorithm that takes advantage of background knowledge in the form of labeled instances to extract the intrinsic density of density-based clusters in a dataset. We have shown that our algorithm outperforms HISSCLU, a hierarchical semi-supervised density-based clustering algorithm when extracting clusters with a single cut, as well as DBSCAN, even when providing it with the best possible global density threshold, using both artificial and real-world data sets. Furthermore, our approach does not require user intervention, as it finds the best cut-values based on the labeled dataset, which makes it a good option to be part of an automatic KDD process.

References

[1] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *ACM SIGMOD International Conference on the Management of Data*, Philadelphia, PA, USA, 1999.

[2] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[3] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. 21st Int. Conf. on Machine Learning, Banff, Alberta, Canada, 2004.*

[4] C. Böhm and C. Plant. HISSCLU: a hierarchical density-based method for semi-supervised clustering. In *Proc. 11th Int. Conf. on Extending Database Technology, Nantes, France*, 2008.

[5] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[6] I. Davidson, K. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. In *Proc. 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany*, 2006.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[8] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[9] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *Proc. 18th International Conf. on Machine Learning*, pages 577–584. Morgan Kaufmann, San Francisco, CA, 2001.

[10] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.