Generalizing from Example Clusters

Pan Hu[1,2], Celine Vens[1], Bart Verstrynge[1], and Hendrik Blockeel[1,3]

[1] KU Leuven, Departement of Computer Science,
Celestijnenlaan 200A, 3001 Leuven, Belgium
[2] Ecole des Mines, Saint-Etienne, France
[3] Leiden Institute of Advanced Computer Science,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract. We consider the following problem: Given a set of data and one or more examples of clusters, find a clustering of the whole data set that is consistent with the given clusters. This is essentially a semi-supervised clustering problem, but it differs from previously studied semi-supervised clustering settings in significant ways. Earlier work has shown that none of the existing methods for semi-supervised clustering handle this problem well. We identify two reasons for this, which are related to the default metric learning methods not working well in this situation, and to overfitting behavior. We investigate the latter in more detail and propose a new method that explicitly guards against overfitting. Experimental results confirm that the new method generalizes much better. Several other problems identified here remain open.

Keywords: Clustering, Semi-supervised Clustering, Constraint-based Clustering, Metric Learning.

1 Introduction

The task of clustering data is ubiquitous in knowledge discovery. Partitional (or non-hierarchical) clustering can be defined as the following task: given a dataset D, partition D into subsets ("clusters") such that instances within the same cluster tend to be similar, and instances in different clusters dissimilar. The notion of "similarity" is crucial here: depending on how this is defined, different solutions will be found. This is true especially for high-dimensional spaces, where different subspaces may reveal different clusterings [1].

It is not always easy for a user to define a good similarity measure. However, users may be able to give examples of instances that in their opinion should, or should not, belong to the same cluster. The clustering system may use this information to understand better the notion of similarity that the user has in mind, and as a consequence produce a better clustering. This type of clustering setting is called semi-supervised clustering, or constraint-based clustering, as the user gives partial information about the desired clustering in the form of constraints that the clustering must fulfill.

Most existing methods for semi-supervised clustering allow the user to provide a number of so-called *must-link* and *cannot-link* constraints, indicating for pairs of instances whether they should (not) be in the same cluster. Vens et al. [11] recently introduced a slightly different setting, called "semi-supervised clustering with example

J. Fürnkranz, E. Hüllermeier, and T. Higuchi (Eds.): DS 2013, LNAI 8140, pp. 64–78, 2013.

clusters". The task can be formulated as follows: given a set of data and one or more examples of clusters in the data, find a clustering of the entire data set that is consistent with these example clusters. While this task could in principle be formulated for hierarchical as well as partitional clustering, we focus here on partitional clustering, as do Vens et al.

This type of supervision is often quite natural. Take, for instance, entity resolution in a database of authors: the task is to cluster occurrences of author names on papers such that occurrences are in the same cluster if they refer to the same actual person.[1] If one person indicates all the papers she authored, that set of papers is an example cluster. Knowing one, or a few, such clusters may help the system determine what kinds of clusters are good, so it can better cluster the other instances. Similarly, when clustering images of faces, one may want to cluster according to identity, poses, emotions, etc. Providing a few example clusters may be easy, and may help the system optimize its similarity measure.

Strictly speaking, the task of clustering from example clusters is a special case of semi-supervised clustering using must-link and cannot-link constraints. Indeed, an example cluster can be translated to a set of such constraints. There are disadvantages to such a translation, however: it can generate a large number of pairwise constraints, and these will be distributed very unevenly over the data set. This is a rather extreme setting for standard semi-supervised clustering methods, and it is not obvious that existing methods can handle it well. In fact, Vens et al. show experimentally that they do not, and also show that a method that explicitly addresses the problem can do better. They do not provide much insight into why this is, though.

In this paper, we analyze the problem of semi-supervised clustering with example clusters in more detail. We provide insight into why existing methods for semi-supervised clustering do not handle this type of problem very well. We identify two reasons. First, most of these methods learn a Mahalanobis distance metric that is more consistent with the given constraints; an alternative view on this is that they implicitly transform the data in such a way that data points that should end up in the same cluster are drawn closer together (and data points that should not, are not). We illustrate visually that these methods may not have the desired effect when the pairwise constraints are concentrated in one cluster. Second, the learned metric has a tendency to overfit the example clusters, especially when learning from few and/or small example clusters, and when learning in high-dimensional spaces. This overfitting was mentioned by Vens et al., but not studied in detail, and no solution was proposed. We here propose an improvement to Vens et al.'s method that explicitly takes the overfitting into account and guards against it. We show that the improved method yields considerably better results.

The remainder of this paper is structured as follows. We first briefly survey the work on semi-supervised clustering (Section 2), including the recent work by Vens et al. [11], on which we build. We next identify two problems that these methods suffer from, and investigate empirically to what extent they do (Section 3). These observations lead to an improved version of Vens et al.'s method, which we describe and empirically evaluate in Section 4. Section 5 presents our conclusions.

[1] This task is not trivial because different persons may have the same name, and the same person may be referred to in different ways, e.g., "John Smith", "J.L. Smith".

2 Semi-supervised Clustering

2.1 Using Pairwise Constraints

Most research on semi-supervised clustering has focused on providing pairwise constraints to the clustering algorithm. In their seminal paper, Wagstaff et al. [12,13] introduce the concept of must-link and cannot-link constraints for specifying, respectively, that two instances should, or should not, be in the same cluster.

One way of dealing with these pairwise constraints is adapting existing clustering algorithms to take them into account. Wagstaff et al. [13] adapt K-Means to this effect. The resulting Constrained Kmeans algorithm iteratively updates cluster centers and cluster assignments exactly like K-Means does, but with the exception that an instance cannot be assigned to a cluster if that would cause constraints to be violated.

Alternatively, one can use a standard algorithm, but adapt the distance metric. Xing et al. [15] propose to learn a Mahalanobis matrix M [8], which defines a corresponding Mahalanobis distance

$$d_M(x,y) = \sqrt{(x-y)^T M(x-y)} \tag{1}$$

They find the M that minimizes the sum of squared distances between instances that must link, under the constraint that $d_M(x,y) \geq 1$ for all x and y that cannot link. M can be restricted to be a diagonal matrix, or can be full. The idea of learning such a distance function is generally referred to as metric-based or similarity-adapting methods [7]. An important point to remember is that the Mahalanobis distance d_M is equivalent to the Euclidean distance in the transformed space obtained by multiplying the data matrix by $M^{1/2}$.

Combining algorithm and similarity adaptation, Bilenko et al. [4] introduced the MPCK-Means algorithm. A first difference with Wagstaff et al. is that constraints are now handled in a soft-constrained manner by defining costs for unsatisfied constraints. Furthermore, the k means are initialized using a seeding procedure proposed by Basu et al. [3]. For metric learning, MPCK-Means provides two options: a separate Mahalanobis metric can be learned for each tentative cluster in every iteration of the algorithm, allowing clusters of different shapes in the final partition (we refer to this as local metric learning), or a single Mahalanobis metric can be learned for the whole space (global metric learning).

2.2 Using Chunklets

Alternatively to pairwise constraints, Bar Hillel et al. [2] use *chunklets*, groups of instances that are known to belong to the same cluster. Their Relevant Component Analysis (RCA) algorithm takes chunklets as input and learns a Mahalanobis matrix. This approach is shown to work better than Xing et al.'s for high-dimensional data. A downside is that only must-link information is taken into account. There is no information about which instances cannot link: different chunklets may belong to the same cluster, or they may not. RCA minimizes the same function as Xing et al.'s method, but under different constraints [2].

Yeung and Chang [16] have extended RCA to include cannot-link information. They treat each pairwise constraint as a chunklet, and compute a separate matrix for the

must-link constraints, A_{ML}, and for the cannot-link constraints, A_{CL}. The data are then transformed by $A_{CL}^{1/2} \cdot A_{ML}^{-1/2}$. This "pushes apart" cannot-link instances in the same way that must-link instances are drawn together. It is equivalent to learning a Mahalanobis distance metric d_M with $M = A_{ML}^{-1/2} \cdot A_{CL} \cdot A_{ML}^{-1/2}$.

2.3 Using Example Clusters

Vens et al. [11] formally define the task of semi-supervised clustering with example clusters as follows ($\mathcal{P}(\cdots)$ denotes the power set):

Given: An instance space X, a set of instances $D \subseteq X$, a set of disjoint example clusters $E \subseteq \mathcal{P}(D)$, and a quality measure $Q : \mathcal{P}(D) \times \mathcal{P}(D) \to \mathbb{R}$.
Find: A partition $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ over D that maximizes $Q(\mathbf{C}, E)$.

Q typically measures to what extent $\mathbf{C}$ is consistent with E (ideally, $E \subseteq \mathbf{C}$), but may also take general clustering quality into account. The number of clusters to be found, or the distance metric to be used, are not part of the input. The requirement that $E \subseteq \mathbf{C}$ is not strict; this allows for noise in the data.

Applications for this setting include all problems where it is easy to provide reasonably accurate example clusters (e.g., entity resolution in author databases, face recognition in picture databases), and where different natural clusterings may exist (e.g., clustering in high-dimensional spaces, where different subspaces may reveal different clusterings, or clustering data that exhibit structure at different levels of granularity).

Example clusters can easily be translated into pairwise constraints, but this results in many constraints (for a single example cluster with n instances in a dataset with N instances, this number is $O(nN)$), and these are highly concentrated (they all involve the example cluster) [11]. This poses problems for several of the existing methods, as we shall see further on. Alternatively, cluster examples can be seen as a special type of chunklets, namely, "maximal" chunklets. RCA can therefore be applied without any translation, but then the information about the maximality of the chunklets is lost (RCA allows separate chunklets to end up in the same cluster, which is not wanted when chunklets are known to be maximal). Yeung and Chang's extension allows for negative information, which solves this problem; but this negative information is again expressed by means of pairwise constraints. Thus, although the chunklet based methods provide a concise representation for the must-link constraints, they do not provide one for cannot-link constraints, so they, too, suffer from the problem of generating many constraints.

Motivated by the above considerations, Vens et al. propose the CLUE algorithm. As we build upon this algorithm, we next discuss it in some more detail.

2.4 The CLUE Algorithm

Algorithm 1 [11] presents CLUE. This algorithm uses a dataset and one or more example clusters as inputs. It does not require the user to provide the number of clusters or a particular distance metric.

Algorithm 1. CLUE

Input: D: a data set, E: a set of example clusters $\{E_i\}_{i=1}^{k}$ with $E_i \subseteq D$
Output: a partition P of D
Algorithm:
1. Rescale all attributes linearly to [0,1]
2. Learn a Mahalanobis distance d_M that is adapted to the constraints
3. Construct a dendrogram Δ by applying a bottom-up hierarchical clustering procedure with d_M
4. Find the range of partitions in Δ for which the examples clusters are reconstructed optimally
5. Within that range, find the best partition P

The algorithm computes a distance metric using an approach similar to that of Yeung and Chang [16]. It computes a Mahalanobis matrix $M = A_{ML}^{-1/2} \cdot A_{CL} \cdot A_{ML}^{-1/2}$, with

$$A_{ML} = \frac{1}{N_a} \sum_{E_i \in E} \sum_{x \in E_i} (x - \bar{E}_i)(x - \bar{E}_i)^T \tag{2}$$

$$A_{CL} = \frac{1}{N_b} \sum_{E_i \in E} \sum_{x \notin E_i} (x - \bar{E}_i)(x - \bar{E}_i)^T \tag{3}$$

where $N_a = \sum_{E_i \in E} |E_i|$ and $N_b = |D| \cdot |E| - \sum_{E_i \in E} |E_i|$.

Thus, while Yeung and Chang use as chunklets the pairwise constraints, CLUE uses as "positive" chunklets the example clusters, and as negative chunklets, pairs $(x, \bar{E}_i)$ with $x \notin E_i$ and $\bar{E}_i$ the mean of the cluster. Intuitively, this pulls instances in the example closer to its center, and pushes other instances farther away from it. The method has complexity $O(n + kN)$, with k the number of example clusters, n the total number of instances in these, and N the overall number of instances, as opposed to $O(nN)$ if Yeung and Chang's method were used.

Next, a standard bottom-up hierarchical clustering method is used, using d_M as distance metric, and using single or complete linkage. In this paper we use the latter. This gives a dendrogram that represents N partitional clusterings, from N singletons at the bottom to a single cluster at the top. This dendrogram is then cut as follows. At each level, the corresponding partition is evaluated using the so-called CORI measure, which tests how well this partition reconstructs the example cluster. With S_{ML} and S_{CL} the set of, respectively, all must-link and cannot-link constraints induced by the example clusters, and $S_{ML}^{correct}$ and $S_{CL}^{correct}$ the sets of all those constraints fulfilled by a clustering C, the CORI of C is defined as:

$$\text{CORI}(C) = \left(\frac{|S_{ML}^{correct}|}{|S_{ML}|} + \frac{|S_{CL}^{correct}|}{|S_{CL}|} \right) / 2 \tag{4}$$

The CORI measure equals 0.5 at the lowest and upper level, and reaches 1 when all example clusters occur in the clustering (i.e., are identical to one C_i). Note that the must-link and cannot-link components of the CORI each have a total weight of 0.5, regardless of how many constraints of each type there are. This is because usually $|S_{CL}| \gg |S_{ML}|$.

The returned clustering is the one with highest CORI. When multiple clusterings have the same highest CORI, the one with best overall clustering quality is chosen, as

measured by category utility [5]. This is an evaluation metric that judges cluster quality in terms of intra- and inter-cluster dissimilarity. Witten et al. [14] proposed a version of category utility for numeric data. CLUE uses "weighted category utility" (WCU), a variant of Witten et al.'s version that takes the Mahalanobis distance into account; it corresponds to implicitly transforming the data according to $M^{1/2}$ and then computing category utility exactly as proposed by Witten et al.

3 Analysis

Our analysis of the behavior of semi-supervised clustering algorithms when learning from example clusters is largely empirical. We first describe the datasets used in the analysis, then the evaluation measures, then we consecutively discuss two weaknesses that the current methods suffer from. In most of this analysis, we assume only one example cluster is given (this is in a sense the most extreme case). The CLUE algorithm is consistently run with Complete Linkage hierarchical clustering [9].

3.1 Datasets

We use the same datasets as in Vens et al. [11]. One is a **synthetic dataset** with 200 instances and 6 numeric dimensions with different domain sizes, see Figure 1. Three dimensions were randomly generated, one dimension contains five bar-shaped clusters, and two dimensions together form 16 circle-shaped clusters. Thus, two possible target clusterings are embedded in these data, which we refer to as **Bars** and **Circles**.

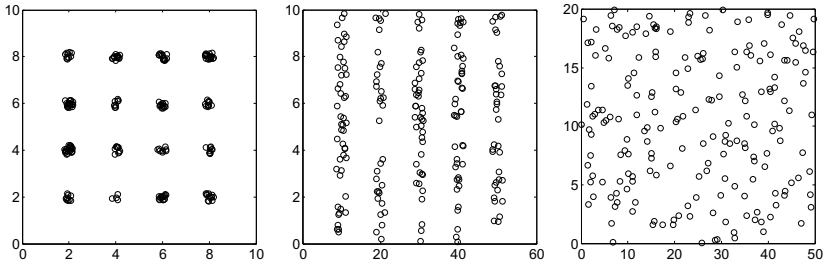

Fig. 1. The 6 dimensions of the synthetic dataset

Besides this, three UCI datasets [6] are used. **CMU Face Images** contains 640 pictures of 20 different persons[2], each shown with 4 poses, 4 emotions, and with or without sunglasses. This is an example of a dataset that can naturally be clustered in several ways. We use the "identity" and "pose" as target clusterings, leading to the **Identity** and **Pose** tasks. Principal Component Analysis was applied to the original data to represent the images as linear combinations of eigenfaces [10]. Only the first 100 eigenfaces were kept; this allowed us to represent the data in a more compact way, while preserving 97% of the original variance in the data. **Libras Movement** contains 15 classes

[2] Due to a corrupted image file, the identity "karyadi" was left out, resulting in 19 identities.

of 24 instances each. Each class references to a hand movement type in Brazilian signal language. **Seeds** contains measurements of seven geometrical properties of kernels belonging to three different varieties of wheat. It has 210 instances.

3.2 Evaluating Clusterings

We use several cluster evaluation measures. In the following, $P = \{p_1, p_2, ..., p_k\}$ denotes the predicted clustering, and $T = \{t_1, t_2, ..., t_l\}$ the target clustering. $c(x)$ denotes the event that an instance x is in a particular cluster c, and $Pr(c(x))$ the probability of this event if x is selected randomly from the data (with c typically of the form p_i or t_j).

The **Rand Index** (RI) is commonly used to compare a predicted clustering to a target clustering. It expresses the proportion of instance pairs for which both clusterings agree on whether they are in the same cluster or not.

When there are many clusters, RI can be dominated by instances correctly predicted not to be in the same cluster. For instance, if $P = \{\{a,b\}, \{c,d\}, \{e,f\}, \{g,h\}, \{i,j\}\}$ and $T = \{\{j,a\}, \{b,c\}, \{d,e\}, \{f,g\}, \{h,i\}\}$, we obtain $RI = \frac{0+35}{45} = 0.778$, a high score for a bad clustering. The **Weighted Rand Index** (WRI) cancels this effect by giving the same importance to the set of pairs that should be in the same clusters, and the set of pairs that should not be (each get weight 0.5). For the clustering given above, we obtain $WRI = \frac{1}{2} \cdot \frac{0}{5} + \frac{1}{2} \cdot \frac{35}{40} = 0.438$

Normalized mutual information (NMI) [9] measures the amount of information that is shared by two clusterings, and penalizes large clusterings. It is defined as follows:

$$NMI(P,T) = \frac{MI(P,T)}{(H(P) + H(T))/2} \tag{5}$$

where MI is mutual information:

$$MI(P,T) = \sum_{i=1}^{k} \sum_{j=1}^{l} Pr(p_i(x), t_j(x)) \cdot \log \left(\frac{Pr(p_i(x), t_j(x))}{Pr(p_i(x)) \cdot Pr(t_j(x))} \right)$$

and H is entropy:

$$H(\{c_1, c_2, \ldots, c_n\}) = -\sum_{i=1}^{n} Pr(c_i(x)) \cdot \log\left(Pr(c_i(x))\right)$$

This measure gives a higher weight to larger clusters, which can be undesirable when cluster sizes may differ substantially. For instance, consider $T = \{\{a\}, \{b\}, \{c\}, \{d,e,f\}\}$ and $P = \{\{a,b,c\}, \{d,e,f\}\}$. If we add more and more instances to the last cluster in both T and P, then $H(P)$ and $H(T)$ will get closer to zero, making P a better clustering, although it still correctly identifies only one of the four clusters in T.

To deal with this, Vens et al. propose a new clustering evaluation measure, called **complemented entropy** (CE). It scores the entropy of the target labels in the predicted clusters (H_t), as well as the entropy of the predicted labels in the target clusters (H_p). These entropies are in a sense complementary: predicting too few clusters increases H_t, predicting too many increases H_p. A formal definition is given below:

$$H_t = -\sum_{i=1}^{k} \sum_{j=1}^{l} Pr(t_j(x) \mid p_i(x)) \cdot \log\left(Pr(t_j(x) \mid p_i(x))\right)$$

$$H_p = -\sum_{j=1}^{l}\sum_{i=1}^{k} Pr(p_i(x) \mid t_j(x)) \cdot \log\left(Pr(p_i(x) \mid t_j(x))\right)$$

$$CE = 1 - \left(\frac{H_t}{maxH_t} + \frac{H_p}{maxH_p}\right)/2 \tag{6}$$

In this definition, $maxH_t$ denotes the highest possible value for H_t, and is reached when all predicted clusters contain an equal number of target labels, and similarly for $maxH_p$.

Whereas NMI gives more weight to clusters with high cardinality, CE gives equal weight to each cluster, irrespective of its cardinality. In the previous example, if we increase the number of instances in the last clusters, CE remains unchanged.

3.3 The Mahalanobis Metric

In this and the following section, we discuss two problems that the current methods for metric learning suffer from, when used with example clusters. First, we show that the learned Mahalanobis metric will not always transform the data in a good way. Second, we show that virtually all metric learning methods overfit the example cluster.

Consider the dataset shown in Figure 2, consisting of four square-like clusters (points drawn from a two-dimensional uniform distribution). Figure 3 shows the transformed instances for two methods that learn a Mahalanobis metric: the method by Xing et al. [15] and the CLUE method [11].

Although the clusters in the original dataset are well-separated, both methods transform the data significantly. By skewing the original clusters, the transformations actually often lead to a worse configuration for clustering, rather than a better one. While in the original dataset, points belonging to different clusters are rarely closer to each other than points drawn from the same cluster, this happens much more frequently in the transformed dataset. Furthermore, the skew that is introduced depends on the example cluster: using a different example cluster results in a very different transformation.

To understand what happens, observe that metric learners tend to push together pairs that must link, and draw apart pairs that cannot link (in the case of Xing et al., these are not drawn apart but a lower bound on their distance is imposed). Assume the lower left cluster is used as an example; then must-link constraints cause compression of the space in mostly random directions, but cannot-link constraints cause expansion of the space mostly to the right, up, or upper right (on average around a line with slope 1). The same holds when the example cluster is the upper right one. For the other two clusters, expansion happens on average around a line with slope -1. This is clearly visible in the figures. Although Xing et al.'s method does not expand the space in this directions, it avoids compression in exactly the same direction, while compressing maximally in the perpendicular direction, so a similar effect is obtained. The one run where no compression is obtained, is where compression is not possible without violating constraints (i.e., the points cannot be projected onto a line without some clusters overlapping).

The effect is demonstrated here for only two methods, but the other methods exhibit similar behavior. The high concentration of must-link and cannot-link constraints in a small area has the effect of stretching the space in the direction where many other points happen to be. There is no reason to believe that this would yield a good metric, unless

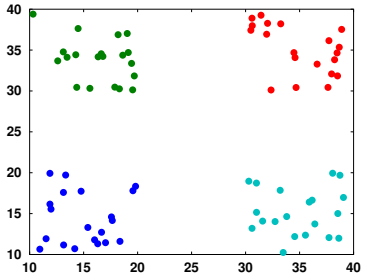

Fig. 2. Synthetic dataset with four clusters

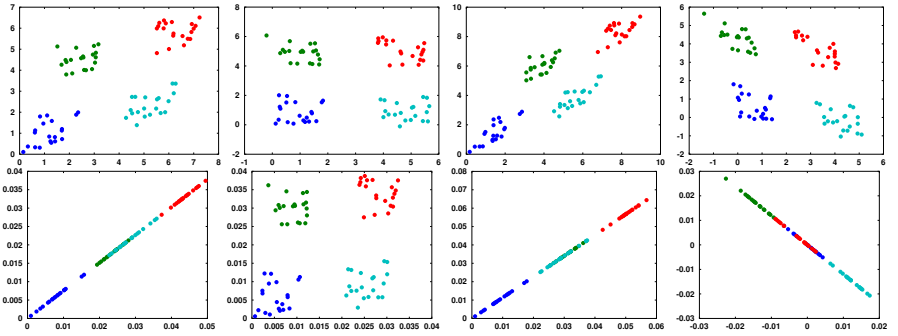

Fig. 3. Transformed data for the synthetic dataset. Metric learning methods: CLUE (row 1) and Xing et al. (row 2). Example clusters: bottom left cluster (column 1), top left cluster (column 2), top right cluster (column 3), bottom right cluster (column 4).

one only wants to separate the one example cluster from all the others. On the contrary, it can have a detrimental effect, as shown here.

3.4 Overfitting

In their earlier experiments with CLUE, Vens et al. observe that it returns too many clusters. It turns out that the CORI measure peaks too early. This can only be explained if example clusters are reconstructed much earlier than other clusters, which means the metric is overfitting the example clusters in the sense that pairs of instances in these clusters are drawn together much more than pairs in other clusters.

That this is indeed the case, is clearly visible in Figure 4, which shows the dendrogram constructed by CLUE for the Libras dataset with a single example cluster. The example cluster is shown in red. It is clear that the metric pushes together the pairs of instances in the example cluster much more than for other clusters, otherwise all clusters should be reconstructed at approximately the same level.

We have measured the degree of overfitting for all methods, as follows. We consider all instance pairs that belong to the same cluster, and sort them according to the learned

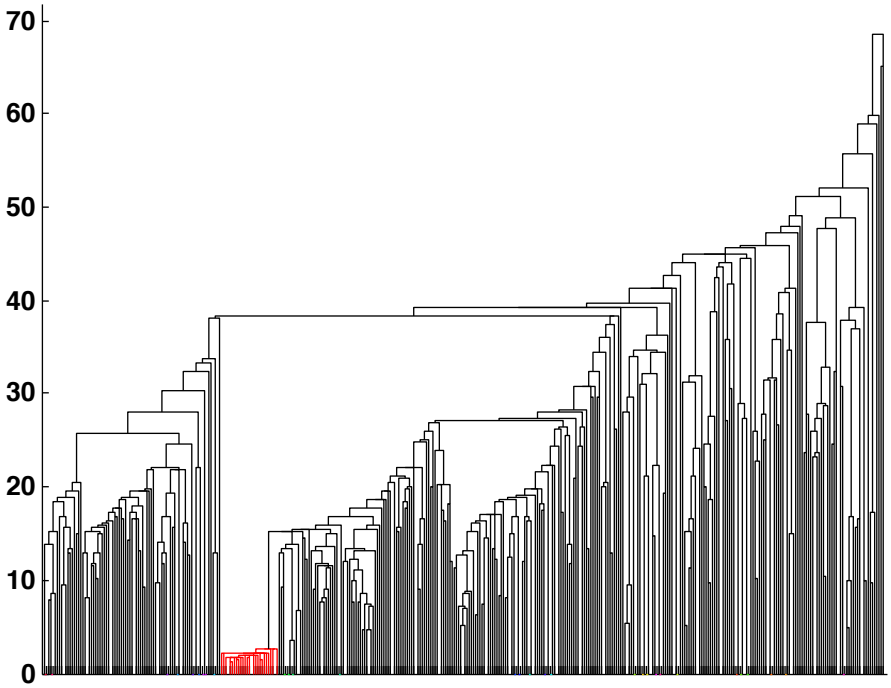

Fig. 4. Dendrogram constructed by CLUE for the Libras dataset. The red part corresponds to the example cluster.

distance between the instances. We then compare the sum of the distance ranks for the instance pairs in the example cluster to the average sum of the ranks over all clusters:

$$\frac{\sum_{x_k,x_l \in E} rank(x_k,x_l)}{\sum_{x_k,x_l \in C} rank(x_k,x_l)/|C|}$$

where E denotes the example cluster, and C denotes any cluster. Table 1 shows the overfitting results for all metric learning methods. We have used each cluster as an example cluster once, and report the average results.

The table clearly shows that, apart from MPCK-Means with global metric learning, all methods have a high risk of strongly overfitting the example cluster. One would expect this risk to be higher as the dimensionality of the space increases (more parameters to be tuned) and the size of the example cluster decreases (fewer independent data to tune these parameters). The table confirms this; for instance, overfitting is worse for Identity (19 small clusters) than for Pose (4 large clusters), which have the same dimensionality, and is worse for Libras (90 dimensions) than for Circles (6 dimensions).

MPCK-Means does not overfit the example cluster, because it does not seek the minimization of distances within the example cluster. Instead, it uses an objective function similar to that of K-Means, but adds penalties for violated constraints, and adapts its metric after each iteration. Thus, metric learning and clustering are interleaved, in

Table 1. Overfitting results. The closer the values are to 1, the less overfitting occurs.

Method	Bars	Circles	Identity	Pose	Libras	Seeds
MPCK-MEANS_global	1.008	0.970	0.996	1.001	0.984	1.000
MPCK-MEANS_local	1.188	1.247	0.725	0.547	0.427	0.939
Xing et al.	0.921	0.674	0.068	0.261	0.150	0.831
RCA	0.869	0.590	0.053	0.268	0.060	0.575
Yeung&Chang	0.899	0.604	0.053	0.274	0.090	0.911
CLUE	0.861	0.564	0.053	0.266	0.080	0.908

Table 2. Comparing distance ranks within and between clusters

Method	Bars	Circles	Identity	Pose	Libras	Seeds
original	0.184	0.076	0.014	0.398	0.041	0.424
MPCK-MEANS_global	0.111	0.027	0.033	0.480	0.041	0.699
CLUE	0.066	0.006	0.011	0.360	0.041	0.431

contrast to the other methods which first learn a metric based on constraints only, then perform clustering. All together, this puts less weight on satisfying the constraints.

The above might suggest that MPCK-Means learns a metric that may fit the example cluster less well, but fits the overall clustering better. As it turns out, this is not the case. We have evaluated this hypothesis as follows.

We consider all instances outside the example cluster, and compute their pairwise distances. We sort these distances, and compare the sum of the distance ranks for pairs of instances in the same cluster to the sum of distance ranks for pairs of instances in different clusters:

$$\frac{\sum_{x_k,x_l \in C} rank(x_k,x_l)}{\sum_{x_k \in C_i, x_l \in C_j, i \neq j} rank(x_k,x_l)}$$

We call this the "within/between measure". Clearly, a good metric will decrease this ratio. Table 2 shows the results for MPCK-Means (with global metric) and CLUE, before and after transformation with the learned metric. We observe that only in 2 out of 6 cases, MPCK-Means improves the rank ratios. For Seeds, Identity and Pose, the results are much worse than without metric learning. CLUE, on the other hand, brings improvements in 4 cases, and has no or a slight opposite effect on 2 cases.

We conclude that MPCK-Means indeed avoids overfitting, yet does not successfully adapt its metric to the clustering problem, whereas the other metric learning methods adapt the metric too much, overfitting the example cluster.

4 CLUE with Defense against Overfitting: CLUEDO

The observation that most systems tend to overfit, and that this is likely due to the pairwise constraints being concentrated in part of the input space, points to an opportunity to improve CLUE. The basic reasoning is as follows. The original CLUE system tends to

return too many clusters because the CORI reaches its optimal point too early, and this happens because overfitting causes the example cluster to be compressed more strongly than the other clusters. Even though the example cluster gets reconstructed early, some pairs from other clusters have already been merged at this point. It seems relatively safe to assume that most of these pairs indeed belong to the same cluster. One could add such merged pairs to the list of must-link constraints and re-run the clustering process.

We have tried several variants of this idea, for the case of a single example cluster.

1. Run CLUE, find the optimal clustering level, add all pairs of examples that got merged before this level to the list of must-link constraints, repeat until some stopping criterion is fulfilled.
2. Run CLUE, find the lowest clustering level for which the example cluster is reconstructed, add all pairs of examples that got merged before this level to the list of must-link constraints, repeat until some stopping criterion is fulfilled.

It turns out both methods suffer from too many erroneous must-link constraints being added, which results in decreasing, rather than increasing, performance. We therefore decided to add must-link constraints more cautiously. This results in the CLUEDO algorithm, shown as Algorithm 2.

Algorithm 2. CLUEDO

Input: D: a data set, E: a set of example clusters $\{E_i\}_{i=1}^{k}$ with $E_i \subseteq D$
Output: a partition P of D
Algorithm:
1. Rescale all attributes linearly to [0,1]
2. For $i = 1$ to 10 do:
   2.1 Learn a Mahalanobis distance d_M that is adapted to the current constraints
   2.2 Construct a dendrogram Δ with a bottom-up clustering procedure using d_M
   2.3 For all instance sets merged below level $iN/10$: add pairwise constraints for all pairs in these sets, unless they violate the original constraints
3. Find the range of partitions in Δ for which the examples clusters are reconstructed optimally
4. Within that range, find the best partition P

Table 3. Overview of CLUEDO results (improvements over CLUE shown in boldface)

		Bars	Circles	Identity	Pose	Libras	Seeds
# Clusters	CLUE	5	22.1	131.5	169.5	254.8	4.3
	CLUEDO	5	**14.9**	**32.1**	**35**	**13.9**	**3**
	true	5	16	19	4	15	3
Overfitting	CLUEDO	**0.89**	**0.97**	**0.70**	**0.38**	**0.69**	**0.95**
Within/between	CLUEDO	0.0664	**0.0054**	**0.0053**	0.3879	**0.0311**	**0.4155**

Table 4. Various evaluation measures for various methods. The last column shows the average rank of the methods. Best results are shown in boldface.

Method	Bars	Circles	Identity	Pose	Libras	Seeds	Avg. rank
			Normalized mutual information (NMI)				
K-MEANS	0.039	0.501	0.717	0.039	0.559	0.641	8.000
CONSTRAINED K-MEANS (CKM)	0.275	0.494	0.738	0.191	0.564	0.747	7.167
MPCK-MEANS_global	0.757	0.281	0.797	0.035	0.535	0.750	6.667
MPCK-MEANS_local	0.391	0.252	0.673	0.026	0.433	0.748	9.000
Xing et al. + CKM	0.889	0.938	0.793	**0.563**	0.548	0.788	3.833
RCA + CKM	0.542	0.631	0.805	0.544	0.580	0.827	4.333
Yeung&Chang + CKM	0.596	0.789	0.775	0.533	0.590	0.879	4.167
CLUE	**1.000**	**0.952**	0.706	0.357	**0.645**	0.453	4.530
CLUEDO	**1.000**	0.934	**0.919**	0.335	0.555	0.660	4.333
CLUEDO + CKM	0.674	0.781	0.885	0.552	**0.645**	**0.883**	**2.667**
			Complemented entropy (CE)				
K-MEANS	0.051	0.529	0.768	0.133	0.615	0.704	8.167
CONSTRAINED K-MEANS (CKM)	0.284	0.519	0.793	0.282	0.603	0.751	7.500
MPCK-MEANS_global	0.757	0.336	0.829	0.074	0.558	0.760	7.167
MPCK-MEANS_local	0.513	0.419	0.711	0.054	0.534	0.764	8.667
Xing et al. + CKM	0.932	0.951	0.840	0.592	0.587	0.792	4.167
RCA + CKM	0.543	0.661	0.845	0.612	0.643	0.835	4.000
Yeung&Chang + CKM	0.602	0.808	0.817	0.590	0.629	0.883	4.667
CLUE	**1.000**	**0.965**	0.780	**0.619**	**0.744**	0.631	3.667
CLUEDO	**1.000**	0.947	**0.940**	0.588	0.633	0.726	3.833
CLUEDO + CKM	0.683	0.800	0.927	0.609	0.672	**0.885**	**3.000**
			Weighted rand index (WRI)				
K-MEANS	0.504	0.587	0.767	0.507	0.662	0.853	8.500
CONSTRAINED K-MEANS (CKM)	0.598	0.590	0.794	0.564	0.664	0.903	6.500
MPCK-MEANS_global	0.838	0.518	0.829	0.515	0.669	0.899	6.000
MPCK-MEANS_local	0.650	0.526	0.730	0.511	0.623	0.891	8.167
Xing et al. + CKM	0.906	0.948	0.832	**0.748**	0.662	0.882	4.167
RCA + CKM	0.717	0.671	0.839	0.734	0.696	0.913	4.000
Yeung&Chang + CKM	0.742	0.798	0.815	0.733	0.692	0.942	4.167
CLUE	**1.000**	0.961	0.658	0.510	0.517	0.679	7.000
CLUEDO	**1.000**	**0.964**	0.894	0.579	0.688	0.835	3.667
CLUEDO + CKM	0.782	0.789	**0.922**	0.744	**0.715**	**0.946**	**2.500**
			Rand index (RI)				
K-MEANS	0.651	0.896	0.930	0.561	0.899	0.853	7.500
CONSTRAINED K-MEANS (CKM)	0.696	0.895	0.928	0.593	0.900	0.903	6.500
MPCK-MEANS_global	0.878	0.866	0.944	0.555	0.888	0.899	6.833
MPCK-MEANS_local	0.670	0.806	0.924	0.554	0.857	0.891	9.000
Xing et al. + CKM	0.918	0.983	0.945	**0.801**	0.897	0.894	3.833
RCA + CKM	0.819	0.918	0.951	0.766	0.895	0.921	4.833
Yeung&Chang + CKM	0.830	0.948	0.946	0.767	0.909	0.948	3.500
CLUE	**1.000**	**0.984**	0.911	0.675	**0.933**	0.679	4.833
CLUEDO	**1.000**	0.975	**0.978**	0.701	0.849	0.835	4.833
CLUEDO + CKM	0.854	0.945	0.961	0.761	0.919	**0.952**	**3.167**

CLUEDO repeatedly runs steps 2 and 3 of CLUE, each time adding more must-link constraints. It first adds the most certain ones, corresponding to pairs that were among the first 10% to get merged (but excluding those pairs that would violate the original constraints). It relearns the Mahalanobis metric with those pairs added, re-clusters with this new metric, then adds also must-link constraints for pairs that got merged in the lower 20% levels. It continues doing this up to the 90% level, then stops and returns the partition with optimal CORI and WCU, as in CLUE.

Table 3 summarizes the results obtained with CLUEDO. It confirms that the number of clusters returned by CLUEDO is often much lower than that of CLUE; also the overfitting metric improves substantially and consistently. This shows that the method achieves its goal of guarding against the overfitting behavior discussed earlier. The within/between measure improves 4 times out of 6, and worsens 1 time.

To put the results into context, we include in Table 4 the NMI, CE, and (W)RI results for a variety of approaches, most of which consist of running constrained K-means with the right number of clusters, and with a metric learned by one of the alternative metric learning methods. Although the evaluation measures generally agree which are the better or worse performing methods, there are some differences. For instance, for Libras, CLUE is the best performing method according to NMI, CE, and RI; however, it is the worst method according to WRI. This results in a lower average WRI rank for CLUE. The results further show that, overall, CLUE and CLUEDO are among the better-performing methods, even though they do not know the number of clusters in advance. Comparing CLUEDO to CLUE, we always observe an improved WRI, while the RI, NMI and CE are more mixed. For completeness, we also included the results for CLUEDO combined with constrained K-means (and hence, using the number of clusters as input) instead of using agglomerative clustering to obtain the final partition. It turns out that this combination has the best average rank over all methods, for all evaluation measures.

5 Conclusions

Generalizing from example clusters is a relatively novel setting for semi-supervised clustering. It can in principle be dealt with by methods that can handle pairwise must-link / cannot-link constraints, but from this point of view it is a rather extreme setting. We have investigated the behavior of these methods in this context, and identified several effects that cause them to perform badly. One is the high concentration of pairwise constraints in one area of the input space, another is the tendency to overfit example clusters. We have quantified these effects for several clustering methods. These observations have led us to propose a new semi-supervised clustering algorithm for this setting, CLUEDO, which is shown to substantially improve upon its predecessor CLUE in terms of finding a clustering with a reasonable number of clusters, and in terms of learning a distance metric that much less overfits example clusters.

Many questions remain. We have mostly focused on extreme situations (generalizing from one example cluster), and while our experiments use a variety of datasets, none of these are very large or high-dimensional. Further, an evaluation of the setting in practical application settings, such as entity resolution (in textual, visual, audio, graph, ... data), would be interesting. Semi-supervised clustering by generalizing from example clusters remains a largely unexplored area that in our view has much potential.

Acknowledgements. C.V. is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO-Vlaanderen). Work supported by the Research Foundation - Flanders (G.0682.11) and the KU Leuven Research Fund (GOA 13/010).

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data. Data Mining and Knowledge Discovery 11(1), 5–33 (2005)
2. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. Journal of Machine Learning Research 6, 937–965 (2005)
3. Basu, S., Banerjee, A., Mooney, R.: Semi-supervised clustering by seeding. In: Proceedings of 19th International Conference on Machine Learning (ICML 2002) (2002)
4. Bilenko, M., Basu, S., Mooney, R.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML, pp. 81–88 (2004)
5. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning 2(2), 139–172 (1987)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
7. Grira, N., Crucianu, M., Boujemaa, N.: Unsupervised and Semi-supervised Clustering: a Brief Survey. A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence, FP6 (2004)
8. Mahalanobis, P.C.: On the generalised distance in statistics. In: Proceedings National Institute of Science, India, pp. 49–55 (1936)
9. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
10. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: Proceedings 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 591(1), pp. 586–591 (1991)
11. Vens, C., Verstrynge, B., Blockeel, H.: Semi-supervised clustering with example clusters. In: Proceedings of the 5th International Conference on Knowledge Discovery and Information Retrieval (accepted, 2013)
12. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1103–1110 (2000)
13. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained K-means clustering with background knowledge. In: ICML, pp. 577–584. Morgan Kaufmann (2001)
14. Witten, I., Frank, E., Hall, M.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2011)
15. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems, vol. 15, pp. 505–512. MIT Press (2002)
16. Yeung, D., Chang, H.: Extending the relevant component analysis algorithm for metric learning using both positive and negative equivalence constraints. Pattern Recognition 39(5), 1007–1010 (2006)