Chapman & Hall/CRC
Data Mining and Knowledge Discovery Series

Constrained Clustering

Advances in Algorithms, Theory, and Applications

Edited by
Sugato Basu • Ian Davidson
Kiri L. Wagstaff

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an informa business
A CHAPMAN & HALL BOOK

Cover image shows the result of clustering a hyperspectral image of Mars using soft constraints to impose spatial contiguity on cluster assignments. The data set was collected by the Space Telescope Imaging Spectrograph (STIS) on the Hubble Space Telescope. This image was reproduced with permission from *Intelligent Clustering with Instance-Level Constraints* by Kiri Wagstaff.

Chapman & Hall/CRC
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-13: 978-1-58488-996-0 (Hardcover)

Library of Congress Cataloging-in-Publication Data

Constrained clustering : advances in algorithms, theory, and applications / editors, Sugato Basu, Ian Davidson, Kiri Wagstaff.
p. cm. -- (Chapman & Hall/CRC data mining and knowledge discovery series)
Includes bibliographical references and index.
ISBN 978-1-58488-996-0 (hardback : alk. paper)
1. Cluster analysis--Data processing. 2. Data mining. 3. Computer algorithms. I. Basu, Sugato. II. Davidson, Ian, 1971- III. Wagstaff, Kiri. IV. Title. V. Series.

QA278.C63 2008
519.5'3--dc22 2008014590

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Chapter 2

Semi-Supervised Clustering with User Feedback

David Cohn

Google, Inc., cohn@google.com

Rich Caruana

Cornell University, caruana@cs.cornell.edu

Andrew Kachites McCallum

University of Massachusetts, Amherst, mccallum@cs.umass.edu

Abstract We present an approach to clustering based on the observation that "it is easier to criticize than to construct." Our approach of *semi-supervised clustering* allows a user to iteratively provide feedback to a clustering algorithm. The feedback is incorporated in the form of constraints, which the clustering algorithm attempts to satisfy on future iterations. These constraints allow the user to guide the clusterer toward clusterings of the data that the user finds more useful. We demonstrate semi-supervised clustering with a system that learns to cluster news stories from a Reuters data set.[1]

2.1 Introduction

Consider the following problem: you are given 100,000 text documents (e.g., papers, newsgroup articles, or web pages) and asked to group them into classes or into a hierarchy such that related documents are grouped together. You are not told what classes or hierarchy to use or what documents are related. Your job is simply to create this taxonomy so that the documents can be browsed and accessed efficiently, either by yourself or by other people. While

[1]This work was originally circulated as an unpublished manuscript [4] when all the authors were at Justsystem Pittsburgh Research Center.

you may have some criterion in mind, you would probably be hard-pressed to express it algorithmically.

This problem is ubiquitous. The web has created a number of new examples of it, but it can be found in many fields that don't involve the web, as well as with many different types of "documents." Librarians, astronomers, biologists — practically everyone tasked with creating a taxonomy from data faces this problem in one form or another.

We propose the following iterative solution to this problem:

1. Give the 100,000 documents to an unsupervised clustering algorithm and have it cluster them.

2. Browse the resulting clusters and tell the system which clusters you like, and which clusters you don't like. Don't do this for all the clusters, just for some of the ones you browsed. Provide feedback to the system by saying *"This document doesn't belong in here," "Move this document to that cluster,"* or *"These two documents shouldn't be (or should be) in the same cluster."*

   Don't do this for all, or even many, of the documents; only for the few that look most out of place.

3. After your critique, re-cluster the documents, allowing the clustering algorithm to modify the the distance metric parameters to try to find a new clustering that satisfies the constraints you provided in the critique.

4. Repeat this until you are happy with the clustering.

This solution is distinct from both traditional supervised and unsupervised learning. Unsupervised clustering takes an unlabeled collection of data and, without intervention or additional knowledge, partitions it into sets of examples such that examples within clusters are more "similar" than examples between clusters. Much work in unsupervised clustering is dedicated to the problem of manually engineering similarity criteria that yield good partitioning of data for a given domain.

Supervised learning, on the other hand, assumes that the class structure or hierarchy already is known. It takes a set of examples with class labels, and returns a function that maps examples to class labels. The goal of supervised learning is to learn mappings that are accurate enough to be useful when classifying new examples, and perhaps to learn mappings that allow users to understand the relationships between the data and the labels, such as which features are important.

Semi-supervised clustering falls between the extremes of totally unsupervised clustering and totally supervised learning. The main goal of our approach to semi-supervised clustering is to allow a human to "steer" the clustering process so that examples can be partitioned into a useful set of clusters with minimum time and human effort. A secondary goal of semi-supervised

clustering is to give the user a way to interact and play with the data so that they can understand it better.[2]

Our approach to semi-supervised clustering assumes that the human user has in their mind criteria that enable them to evaluate the quality of a clustering. It does not assume that the user is conscious of what they think defines a good clustering but that, as with art, they will "know it when they see it." Most importantly, semi-supervised clustering never expects a user to write a function that *defines* the clustering criterion. Instead, the user *interacts* with the clustering system, which attempts to learn a criterion that yields clusters the user is satisfied with. As such, one of the primary challenges of semi-supervised clustering is finding ways to elicit and make use of user feedback during clustering.

The remainder of this chapter describes one simple, illustrative way in which this may be accomplished. Other challenges that need to be addressed by future research on semi-supervised clustering are briefly described in the discussion section.

2.1.1 Relation to Active Learning

Semi-supervised clustering with user feedback is closely related to active learning [5]. In the most common form of active learning, a learning system attempts to identify which data points, if labeled by a human, would be most informative. In semi-supervised clustering, the *human* selects the data points, and puts on them a wide array of possible *constraints* instead of labels. These two key differences point toward some situations in which the semi-supervised approach is preferable.

1. In some clustering problems the desired similarity metric may be so different from the default that traditional active learning would make many inefficient queries. This problem also arises when there are many different plausible clusterings. Although less automated, a human browsing the data would do less work by selecting the feedback data points themself.

2. The intuitive array of possible constraints are easier to apply than labels, especially when the final clusters are not known in advance.

3. The very act of human browsing can lead to the discovery of what clusters are desired. Semi-supervised learning can thus be seen as a method of data exploration and pattern discovery, efficiently aided by cluster-based summarization.

[2] Demiriz et al. [7] independently introduced a semi-supervised clustering model similar to the one we describe here. The main distinction between our work and theirs is our use of iterative feedback to acquire labelings; Demiriz et al. assume that all available labels are given a priori.

However, the distinction with active learning is subjective. As we will see in Section 2.5.1, our system could easily be viewed as a practical application of learning by counterexamples [1] – one of the earliest and most powerful forms of active learning studied in the theory community.

Hybrid active-semi-supervised systems are also plausible. In situations with a large number of data points and data types that are difficult to browse, one could imagine a system that combines some of the automated selection of active learning with the human browsing of semi-supervised clustering. The active learner could make many disparate hypotheses about the underlying labels and present the examples that would be most indicative of each.

2.2 Clustering

Formally, clustering is the process of partitioning a data set into subsets such that all members of a given subset are "similar" according to some distance measure D. We will denote the distance between two examples x_1 and x_2 as $D(x_1, x_2)$. We can generalize this to refer to $D(y_1, y_2)$, the distance between two cluster centers, or $D(y_1, x_1)$, the distance between a cluster center and an example.

The two most popular approaches to clustering are agglomerative clustering and prototype-based clustering. In agglomerative clustering, each datum is initially placed in its own cluster. The clusters that are most similar (according to D) are iteratively merged, until the desired number of clusters is reached, or some limit on data likelihood or distortion is exceeded (see Hofmann and Buhmann [14] for an in-depth treatment of agglomerative clustering).

In prototype-based clustering, the final number of clusters is usually set a priori, and the corresponding prototypes are found using some form of Expectation Maximization (EM) [8]. Each prototype is initialized to some position (in our case, a randomly weighted sample of the training points). Examples are assigned to prototypes according to their similarity to each prototype (the assignment may be 0-1 or fractional, depending on the algorithm). Prototypes are then adjusted to maximize the data likelihood, or, equivalently, minimize the data distortion. The assignment/adjustment process is repeated until no significant changes result (see Meilă and Heckerman [17] for concise review of prototype-based clustering).

In the present chapter, we adopt a statistical prototype-based approach, resulting from the naive Bayes model of document generation [16].[3] Given a

[3]We reiterate that the approach described in this chapter is only for the point of exploration and illustration; the approach is, in theory, applicable to almost any clustering algorithm.

vocabulary V, a document is assumed to be a "bag of words" generated from a multinomial distribution θ. In this model, the probability of document x is

$$P(x) = \prod_{t_j \in V} P(t_j|\theta)^{N(t_j,x)},$$

where $P(t_j|\theta)$ is the parameterized probability of term t_j being generated, and $N(t_j, x)$ is the number of times t_j appears in the document. Each document x forms an estimate of a multinomial distribution θ_x; likewise, each cluster of documents π forms an estimate θ_π composed from the θ_x of its constituent documents.[4]

For clustering we assume that, instead of being produced by a single multinomial distribution, each of the observed documents was drawn from one of distributions $\theta_{\pi_1}, \theta_{\pi_2}, \ldots, \theta_{\pi_k}$, corresponding to the unknown distribution of clusters $\pi_1, \pi_2, \ldots, \pi_k$:

$$P(x) = \sum_i P(\pi_i)P(x|\pi_i) = \sum_i P(\pi_i) \prod_{t_j \in V} P(t_j|\theta_{\pi_i})^{N(t_j,x)}.$$

Our task is to estimate values for $P(\pi_i)$ and θ_{π_i}, which will in turn allow us to estimate cluster memberships $P(\pi_i|x)$ by Bayes rule:

$$P(\pi_i|x) = P(x|\pi_i)P(\pi_i)/P(x). \tag{2.1}$$

We find estimates for $P(\pi_i)$ and θ_{π_i} via the standard procedure for EM, beginning with randomized estimates of θ_{π_i} drawn as a weighted sample from the observations. Then, for each cluster π_i and document x, we compute $P(x|\theta_{\pi_i})$ and apply Equation 2.1 to compute $P(\pi_i|x)$. Each cluster is given partial ownership of a document proportional to $P(\pi_i|x)$. The parameters θ_{π_i} are recomputed as the weighted sum of their component documents, and the process is repeated. The algorithm is guaranteed to converge to a locally optimal clustering (see, e.g., MacKay [15] or Meilă and Heckerman [17] for details).

2.3 Semi-Supervised Clustering

The goodness of any clustering depends on how well the metric D matches the user's (perhaps unknown) internal model of the target domain. We propose allowing the user to impose their model on the metric via the clustering

[4]The estimates for term probabilities are derived from the relative term frequencies in the documents. Following McCallum and Nigam [16], we smooth with a LaPlacean prior to avoid zero term probabilities.

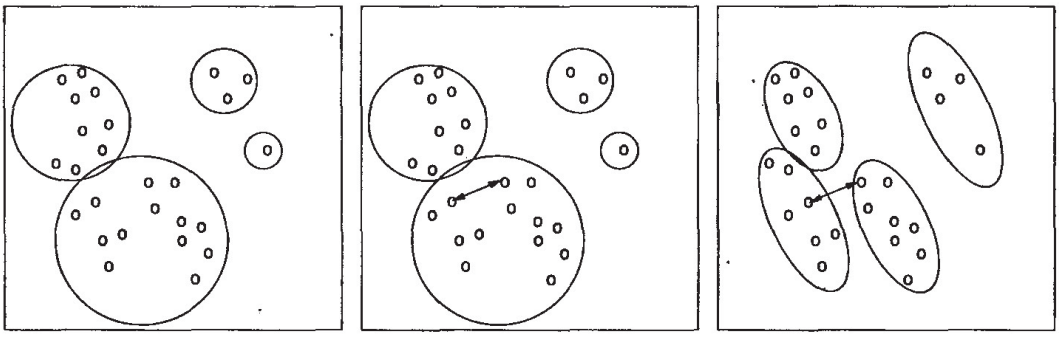

FIGURE 2.1: Illustration of semi-supervised clustering. Given an initial clustering, the user specifies two points that should not have been placed in the same cluster. The system warps its metric, allowing it to find a clustering that respects the constraint.

algorithm, by having the user provide the algorithm with feedback, and allowing it to alter the metric so as to accommodate that feedback. Not only is it easier to critique than to construct, but the user's criticism can take many forms — specifying that a particular example does/doesn't belong in a particular cluster, that two examples do/don't belong in the same cluster, or that a particular cluster is good (and should be preserved) or bad (and should be split up).

Feedback may be incorporated into the metric as constraints to be respected by the clustering algorithm. Consider two examples, x_1 and x_2, that are constrained by the user feedback to be in separate clusters. When the clustering algorithm attempts a partitioning which places x_1 and x_2 in the same cluster, the metric may be altered to increase the distance between x_1 and x_2 until one or the other of them falls in a different cluster (Figure 2.1). Other constraints may be implemented similarly, shrinking the distance between some example and a cluster prototype, or increasing the distance between a cluster prototype and all the examples assigned to it.

2.3.1 Implementing Pairwise Document Constraints

In this probabilistic setting, the natural measure of dissimilarity between two documents, x_1 and x_2, is the probability that they were generated by the same multinomial. From Pereira et al. [19], this is proportional to the *KL divergence to the mean* of their multinomial distributions:

$$D_{KLM}(x_1, x_2) = |x_1| D_{KL}(\theta_{x_1}, \theta_{x_1,x_2}) + |x_2| D_{KL}(\theta_{x_2}, \theta_{x_1,x_2}),$$

where $|x|$ is the length of document x, $D_{KL}(\theta_1, \theta_2)$ is the standard Kullback-Leibler divergence of θ_1 to θ_2, and θ_{x_1,x_2} is a distribution such that

$$P(t_j|\theta_{x_1,x_2}) = \left(P(t_j|\theta_{x_1}) + P(t_j|\theta_{x_2})\right)/2.$$

The advantage of this measure is that it is symmetric, unlike standard KL divergence.

To implement our constraints, we augment the standard KL divergence $D(\theta_{x_1}, \theta_{x_2})$ with a weighting function

$$D'_{KL}(\theta_{x_1}, \theta_{x_2}) = \sum_{t_j \in V} \gamma_j \cdot P(t_j|\theta_{x_1}) \log \left(\frac{P(t_j|\theta_{x_2})}{P(t_j|\theta_{x_1})} \right).$$

where γ_j may be interpreted as indicating the importance of t_j for distinguishing x_1 and x_2. Then, given a constraint that x_1 and x_2 must be in separate clusters, we can warp the metric by computing

$$\frac{\partial D'_{KLM}(x_1, x_2)}{\partial \gamma_j} = |x_1| P(t_j|\theta_{x_1}) \log \left(\frac{P(t_j|\theta_{x_1 x_2})}{P(t_j|\theta_{x_1})} \right) + |x_2| P(t_j|\theta_{x_2}) \log \left(\frac{P(t_j|\theta_{x_1 x_2})}{P(t_j|\theta_{x_2})} \right)$$

and hillclimbing over γ to increase the effective distance between the two. This gradient tells us the direction to move the γ's in order to increase (or decrease) the separation between two documents. (In the current experiments we constrain the γ's to be positive, but it might be interesting to relax this and allow some γ's to become negative.)

These γ's are incorporated back into the E-step of clustering algorithm as weights attached to the individual term frequencies:

$$P(x|\pi_i) = \prod_{t_j \in V} P(t_j|\theta_{\pi_i})^{\gamma_j N(t_j, x)}.$$

Intuitively, a small γ_j reduces the effect of t_j's presence or absence on document likelihood, effectively scaling its effect on the document's divergence from its cluster center. As such, we are able to inject a learned distance metric directly into the clustering algorithm.

2.3.2 Other Constraints

Other constraints described in the previous section may be similarly implemented by hillclimbing over the example-to-cluster and cluster-to-cluster distance. Note that the linear warping we describe will not guarantee that all constraints can be satisfied; some clusterings desired by the user may be non-convex and unrealizable in the space of models supported by naive Bayes. In this case, the hillclimbing will converge to a weighting that provides a local minimum of constraint violations. Local or nonlinear warpings of the distance metric, such as the ones described by Friedman [11] and Yianilos [21] may be of use in these situations.

2.4 Experiments

In this section, we illustrate the semi-supervised approach on a small document clustering problem. We use a set of 25 documents each from five Reuters topic areas: business, health, politics, sports, and tech. Starting from five randomly initialized prototypes, the EM-based clustering algorithm described in the previous sections finds clusters that maximize data likelihood.

Each time clustering converges, we add a constraint. We simulate a human user by identifying two documents from the same cluster whose sources are different Reuters topics, and constrain them to be in different clusters.[5] For each unsatisfied constraint, we reweight the divergence by a fixed number of hillclimbing steps, re-initialize the cluster prototypes, and repeat the EM training.

2.4.1 Clustering Performance

Figure 2.2 compares the performance of supervised, unsupervised, and semi-supervised learning. For unsupervised and semi-supervised learners, we plot cluster purity: the fraction of examples that would be classified correctly if all examples were assigned the majority label in each cluster. For the supervised learner, we plot both cluster purity and classification accuracy (generalization).

After only a few constraints have been added, cluster purity increases sharply over that of unsupervised clustering. It is not clear, however, how to fairly compare the performance of semi-supervised clustering with that of fully supervised clustering: constraints do not exactly correspond to labeled examples, and it is uncertain what constitutes a proper test set. In supervised learning, documents used for training are traditionally excluded from the test set, since their labels are already known. But the semi-supervised model clusters (and is tested on) the entire corpus, so it is also reasonable to gauge it against a supervised learner tested the same way. In the figure we show the cluster purity of supervised learning on the training set as well as its generalization to an independent test set.

The semi-supervised learner reaches its asymptotic performance after about 10 constraints have been added; the supervised learners require between 3 and 6 times more labeled examples to reach that level of performance.[6] It is in-

[5] A fully-operational semi-supervised clustering system would benefit from a graphical user interface that permits efficient browsing of the current clusters and supports easy specification of user constraints. See the discussion of Scatter/Gather later in this chapter.

[6] To assure ourselves that metric-warping alone wasn't responsible for the performance disparity, we also incorporated metric warping into the supervised clusterer, shrinking the divergence between a document and its assigned cluster. The addition resulted in no significant performance improvement.

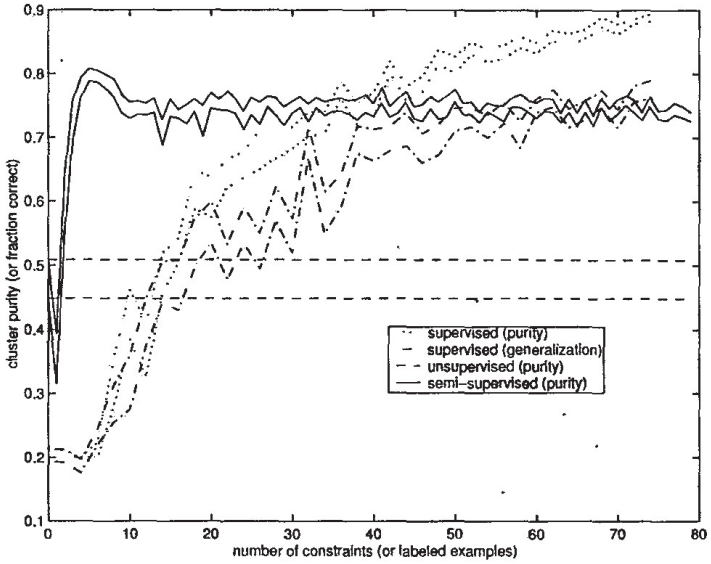

FIGURE 2.2: Learning curves for supervised, unsupervised, and semi-supervised clustering. For supervised clustering, cluster purity (measured on the train set) and generalization (measured on an independent test set) are plotted against the number of labeled examples; for semi-supervised clustering, purity is plotted against the number of constraints. Averages over 10 runs each, with the upper and lower lines indicating error bars at one standard deviation. See text for details.

teresting to note that the performance of the semi-supervised learner actually begins to decrease after roughly 20 constraints have been added. The Reuters data set contains many documents that appear under more than one topic (an identical article on Microsoft, for example, appears under both *business* and *tech*). We hypothesize that, in an attempt to separate these unseparable documents, the learner is pushing its term weightings to unhealthy extremes.

Experiments on a larger data set consisting of 20,000 USENET articles suggest that semi-supervised clustering is just as effective with large data sets. More importantly, these experiments show that semi-supervised clustering is able to cluster the same data according to different orthogonal criteria. This data set contains articles on four subjects: aviation simulators, real aviation, auto simulators, and real autos. Semi-supervised clustering can cluster the simulators and real groups together (e.g., aviation simulators and real aviation) or the auto and aviation groups together (e.g., aviation simulators and auto simulators) depending on the feedback provided by the user. In both cases it does so at about 80% accuracy with 10 constraints. When the distance metric is not adjusted, the same constraints give an average of only 64% accuracy. (Purely unsupervised clustering achieves only about 50% accuracy.)

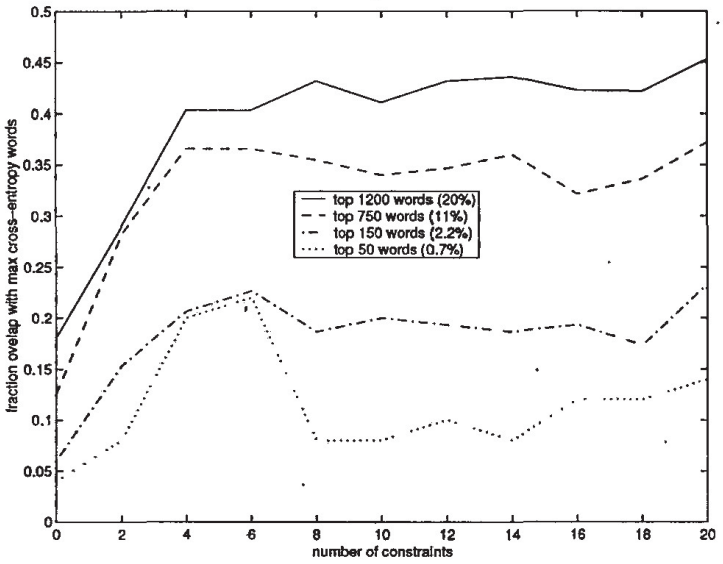

FIGURE 2.3: Fraction overlap of the top n weighted terms with top n terms ranked by information gain on fully-supervised data. As the number of constraints increases, there is increasing correlation with terms that strongly affect class conditional probabilities. Note that this overlap is achieved with far fewer constraints than the number of labels in the fully-supervised data.

2.4.2 Learning Term Weightings

Adjusting γ_j warps the metric by adjusting the resolving power of term t_j, essentially identifying which terms are most useful for distinguishing documents. If γ_j is large, small disparities in the frequency of t_j become important and will tend to separate documents; if γ_j is small, large disparities in frequency will be ignored.

Empirically, this behavior is borne out on the Reuters experiments. Terms that subjectively appear highly relevant for distinguishing topics, such as *Iraq*, *economy*, *weapons* and *council* are given large weightings. We computed the information gain of t_j using all document labels [18], and compared it with γ_j. Figure 2.3 shows the overlap between the top-weighted $n\%$ terms in the vocabulary with the same terms ranked by information gain. After about a dozen constraints, semi-supervised clustering learns term weightings with moderate overlap to the term weightings learned by supervised learning from all 125 document labels.

2.5 Discussion

This chapter only scratches the surface of semi-supervised clustering with user feedback. There are still many issues to be addressed; we touch on a few of these next.

2.5.1 Constraints vs. Labels

When applying supervised learning to classification problems, it is assumed that the users know the target classes and have labeled examples from each target class. In many interesting problems, this is an unrealistic assumption. A semi-supervised system allows users to give label-like information to the learner without having to know labels. Although user feedback in semi-supervised clustering serves a similar role as class labels serve in supervised learning, comparing supervised learning with semi-supervised clustering is an apples–to–oranges comparison. Semi-supervised clustering usually will be applied to problems where labels are not readily available. However, evaluating clustering systems is difficult and usually subjective. We compare the performance of semi-supervised clustering to supervised learning using a labeled data set principally to avoid this subjectivity.

The performance disparity between supervised and semi-supervised clustering is surprising. While we have argued that it is easier to provide constraints than labels, constraints also provide less information than labels. Constraints don't require the user to know the correct label (or even what labels exist!) — only the relationship among pairs or sets of labels. There are only 125 possible labels in the small Reuters data set, but thousands of possible separation constraints. Yet empirically, even with very few constraints, the semi-supervised learner is able to perform surprisingly well.

One explanation is in the connection to active learning. As a means of user feedback, the addition of a constraint indicates a problem and effectively acts as a counterexample for the present clustering. Counterexamples are a powerful tool for doing active learning, which, in some situations, are much more efficient than learning from randomly labeled examples [1]. As such, the user, by iteratively directing the clusterer's attention toward points that are incorrectly clustered, gives a semi-supervised clustering system the many advantages of an active learning system.

2.5.2 Types of User Feedback

As we have discussed, there are many different types of feedback that users might provide to a semi-supervised clustering system. One type of feedback is the constraints on individual data points and clusters we used earlier. But many other forms of feedback might prove useful as well. For example, a user

might tell the system that the current clustering is too coarse or too fine. Or the user might point to a cluster and indicate that the cluster is bad without saying how it is bad. Similarly, a user might indicate that a cluster is good, suggesting that future re-clusterings of the data should attempt to maintain this cluster. Users might also give feedback that is not cluster specific, such as telling the system that the entire clustering looks bad and that the next clustering should be very different.

Some types of user feedback may require adaptive clustering that cannot be easily handled by the γ weighting scheme we used above. For example, we considered an approach to finding good—but qualitatively different—clusterings of the same data by exploiting EM's *weakness* for getting trapped in local minima. Different local minima may capture qualitatively different ways of clustering the data, one of which may better match the user's internal preference function than the deepest minima the system can find. In the long run we hope to develop a general framework for representing user feedback about clusters.

2.5.3 Other Applications

We believe there are many applications of feedback-driven semi-supervised clustering. Imagine a Yahoo! hierarchy for web pages that allows the user to tailor the hierarchy to better match their own interests by providing feedback while browsing. Similarly, consider an automatic e-mail system in which a user allows the system to cluster e-mail into related mailboxes instead of manually specifying the mailboxes. Semi-supervised feedback would allow the user to tailor mailbox clusters to fit their (possibly changing) needs. As a different example, consider a user clustering proteins into homology groups (groups of proteins with similar structures). Large proteins have complex structures and could be clustered many different ways. A feedback-driven semi-supervised clustering system would allow the user to explore many different ways the proteins might be clustered and to find clusterings most suitable to their purposes.

2.5.4 Related Work

The core operation of semi-supervised clustering involves learning a distance metric, of which a great deal of work has been done for classification problems (see Hastie and Tibshirani [13] for an overview); more recently, researchers have begun applying these techniques to clustering and other forms of machine learning (see, e.g., Xing et al. [20]).

As indicated earlier, our model is most similar to the work of Demiriz et al. They report how a fixed set of labeled examples may be used to bias a clustering algorithm; we investigate how a user, interacting with the system, may efficiently guide the learner to a desired clustering.

In the time since this work was first presented, there has been a great deal

of research in improving clusterings by the (semi-supervised) learning of a distance measure. Instead of attempting a complete list of references here, we refer the reader the references in Chapter 1 and to the other, more recent contributions in this volume.

Our technique of incorporating user feedback is a cousin to relevance feedback, a technique for information retrieval [2]. Given a query and initial set of retrieved documents, relevance feedback asks the user to tag documents as being more or less relevant to the query being pursued. As the process is iterated, the retrieval system builds an increasingly accurate model of what the user is searching for.

The question of how a user (or teacher) may best select examples to help a learner identify a target concept is the focus of much work in computational learning theory. See Goldman and Kearns [12] for a detailed treatment of the problem.

The Scatter/Gather algorithm [6] is an interactive clustering algorithm designed for information retrieval. The system provides an initial clustering of data. When the user selects a subset of the clusters for further examination, the system gathers their components and regroups them to form new clusters. Scatter/Gather aims at pursuing and finding structure in a small part of a corpus. This makes it an interesting complement to our approach: Scatter/Gather may provide an effective means for browsing and focusing on clusters of interest, and semi-supervised learning may be an effective means of improving the quality of those clusters.

Note that we do not compare our performance to that of other purely unsupervised clustering systems such as AutoClass [3], COBWEB [9], or Iterative Optimization [10]. The contribution of our work is not to introduce a new clustering *algorithm*, but an approach that allows user feedback to guide the clustering. While we have illustrated our approach on a relatively simple system, we believe it is equally applicable to more sophisticated algorithms, and expect that it will provide similar improvements over the unsupervised variants.

References

[1] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[2] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In *Proceedings of the 18th Annual International Association for Computing Machinery (ACM) Special Interest Group on Informa-*

tion Retrieval Conference on Research and Development in Information Retrieval, pages 351–357. ACM Press, 1995.

[3] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A Bayesian classification system. In *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, pages 431–441. Morgan Kaufmann, 1993.

[4] David Cohn, Rich Caruana, and Andrew McCallum. Semi-supervised clustering with user feedback. Unpublished manuscript (later released as Cornell University Technical Report TR2003-1892), 1999.

[5] David Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[6] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International Association for Computing Machinery (ACM) Special Interest Group on Information Retrieval Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[7] A. Demiriz, K. P. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of Artificial Neural Networks in Engineering*, 1999.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B (Methodological)*, 39(1):1–38, 1977.

[9] Doug H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[10] Doug H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4:147–180, 1996.

[11] Jerome Friedman. Flexible metric nearest neighbor classication. Technical Report 113, Stanford University, Department of Statistics, 1994.

[12] Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

[13] Trevor Hastie and Rob Tibshirani. Discriminant adaptive nearest neighbor classification. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Pattern Analysis and Machine Intelligence*, 18:607–616, 1996.

[14] Thomas Hofmann and Joachim M. Buhmann. Pairwise data clustering by deterministic annealing. *Institute of Electrical and Electronics*

Engineers (IEEE) Transactions on Pattern Analysis and Machine Intelligence, 19(1):1–14, 1997.

[15] David J.C. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[16] Andrew McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. In *Workshop on Learning for Text Categorization at the 15th Conference of the American Association for Artificial Intelligence*, 1998.

[17] Marina Meilă and David Heckerman. An experimental comparison of several clustering and initialization methods. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI 98*, pages 386–395. Morgan Kaufmann, 1998.

[18] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[19] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics, 1993.

[20] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

[21] Peter N. Yianilos. Metric learning via normal mixtures. Technical report, NEC Research Institute, 1995.