DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **70**, 2006

Some Key Concepts in Data Mining – Clustering

Graham Cormode

1. Introduction

The notion of 'clusters' is a very natural one, and occurs frequently in discussions of epidemiology. We hear about 'cancer clusters', areas where the number of reported cancer cases within an area or group of people exceeds the expected amount. Such clusters lead to investigation of possible carcinogens or explanation for greater susceptibility amongst certain groups.

When thinking about clusters, people most often think of geographical clusters—the image of "pins in a map" is a resonant one. In fact, this approach has its roots in the very foundation of epidemiology. The famed work of John Snow in 1854 was to plot cases of cholera on a map of London, and to observe that these were centered around certain water pumps. Thus the result of this early instance of clustering was to produce a hypothesis on the source of the cholera cases, which was subsequently verified. The evolution of this early success in visualizing data has been the development of sophisticated GIS systems, which can rapidly plot a variety of data on top of maps in many ways.

However, not all data is geographic in nature. Patient admission records contain large numbers of variables of different types: geographic (home address, work address); time (date of birth, duration of symptoms); categorical (existing medications, socio-economic indicators); textual (physician's notes); and numeric (patient readings, such as heart rate, blood pressure etc.). We would also like to cluster such data in order to find patterns of similarity that can lead to new hypotheses. It is no longer possible to plot such high dimensional and heterogeneous data on a chart and visually pick out clusters. Instead, we turn to data mining approaches to take the data and find the clusters therein.

We will focus on the data mining methods used to produce these clusters, but there are many other aspects of the problem surrounding this that we only touch on.

- **Data Collection.** Data Collection is a major hurdle for any data mining task. Designing experiments and enrolling volunteers to get a significant study group can be a major commitment. Sometimes data mining is characterized as "data dredging": taking existing data sets collected for some other purpose and 'dredging' them for new information. Even this approach is fraught: since the data was not collected with the current goal

in mind, there may be relevant attributes missing. Important information about the collection method may have been lost. It may be necessary to fuse together multiple data sets, and done without sufficient care this can lead to erroneous conclusions. Lastly, there are myriad issues of privacy and consent: subjects may have agreed to their data being used in one study, but do not want it to be used for innumerable others. Issues of individual privacy also mean that some attribute values (home address, date of birth) may be omitted, randomly perturbed by a small amount, or otherwise changed. Data users need to be aware of all these effects before drawing conclusions from the data.

- **Data Cleaning.** No data set is perfect. At the very least, one can expect missing values for some attributes, some errors in transcription or data input, and duplicate entries. Dealing with these issues is a topic of major study in itself [**1**]. Sometimes, a received data set has already been 'cleaned'. Perhaps 'scrubbed' is a better term: missing values are sometimes filled in with average values, or values copied from similar looking records. Values outside "sanity bounds" (ages greater than 120, pulse rates below 0) may be replaced with default values, or the corresponding record dropped completely. As with data collection, it is important to know the methodology applied to clean the data in order to interpret conclusions correctly.
- **Interpreting Results.** Most data mining tasks do **not** give as their output a set of hypotheses about diseases and their cause. Instead, they merely produce observations about the input data, to some degree of confidence, and it is up to the user to draw their own conclusions. To return to a geographical example, we might plot cancer data across the United States, and observe much higher incidence in some areas, such as Florida. Before forming hypotheses about possible carcinogens prevalent in these areas, we need to consider to what extent this can be explained by already known factors. In this case, if we had not adjusted the data for demographic factors, then the observation may be explained by the fact that many senior citizens retire to Florida, and incidence of cancers increase with age. Great care must be taken to adjust for all relevant factors before claiming that observed results are significant, and frequently subsequent work is needed to verify initial findings, and ensure that the input data was not anomalous.

We continue our discussion of clustering as follows. In the next subsection we will give a mathematical formalism to clustering, and define it as an optimization problem over input data. Then we consider three popular clustering methods: hierarchical clustering, the k-means algorithm, and Expectation Maximization (EM).

2. The Clustering Problem

In order to cluster the input into sets of similar points, we need to be able to define a distance between any pair of points to gauge their similarity. Formally, we assume that the input is a set of points from a *metric space*, with an associated metric, or distance, function. We will denote this distance as d, so the distance between two points x and y is given by $d(x, y)$. If the points are in a metric space, then this gives three requirements on d:

(1) **Identity:** $d(x, x) = 0$ — the distance from any point to itself is zero.
(2) **Symmetry:** $d(x, y) = d(y, x) \geq 0$ — the distance between any two points is the same in both directions, and is non-negative.
(3) **Triangle Inequality:** $d(x, z) \leq d(x, y) + d(y, z)$ — this means that it is never quicker to get from one point to another by going via a different point.

In extensions of clustering applications, it is possible to drop some or all of these conditions, but we will focus on the metric space setting.

Defining an appropriate distance function can be challenging. If all the data is numeric, then we can use the Euclidean distance function (straight line distance) or L_∞ distance (maximum distance in any co-ordinate). However, real data is rarely like this. We could map our data onto numerical values, although the choice of scale can affect the result dramatically. Suppose we had a categorical attribute which takes two values: malign, and benign. We could map benign to 0, and malign to 1, or to 0 and 100, respectively. The choice of this difference effectively determines the extent to which we are emphasizing that attribute relative to the others. One can attempt to normalize the data, by rescaling all values into the range $[0 \dots 1]$, but this still does not solve the problem completely. Once the distance function has been chosen, we can go on to define the general clustering problem.

Definition 1 (Clustering Algorithm). *A clustering algorithm takes as input a set of points from a metric space, and outputs a set of clusters, $\mathcal{C} = \{C_1 \dots C_k\}$.*

Note that this definition is very broad—it does not describe how the clusters are described or what criteria they fulfil. This is because there are many ways to define the desired clustering. We give two commonly used formalisms:

Definition 2 (k-center). *A k-center clustering algorithm outputs a set of k points $\mathcal{C} = \{C_1 \dots C_k\}$ ("centers") from the metric space to define the clusters: each input data point is associated with the point from $\mathcal{C}$ that is closest to it (ties can be broken arbitrarily). The quality of the clustering is determined by the maximum distance of a point to its closest center, ie $\max_x \min_i d(x, C_i)$.*

Definition 3 (k-median). *A k-median clustering algorithm outputs a set of k points $\mathcal{C}$ ("medians") from the metric space to define the clusters: each input data point is associated with the point from $\mathcal{C}$ that is closest to it (ties can be broken arbitrarily). The quality of the clustering is determined by the average distance of points to their closest median, ie for n points this is $\frac{1}{n} \sum_x \min_i d(x, C_i)$.*

Note that in both cases, the way points are allocated to clusters is the same, but it is the *objective function* that varies—that is, the function that we wish to minimize to get the best clustering. To give a good clustering we want to find centers/medians so that the generated clusters give a good covering of the points. This definition also puts each point in exactly one cluster; more general definitions allow a point to belong to multiple clusters, perhaps with some degree of certainty/probability.

Formally speaking, both these objectives are *NP-Hard* to optimize. That is, all known algorithms to find the optimal solution take time exponential in the size of the input. In theoretical computer science, researchers look for algorithms with guaranteed approximation factors, that come provably close to the optimal solution. However, these tend to be complicated and are often slow in practice. Instead, we

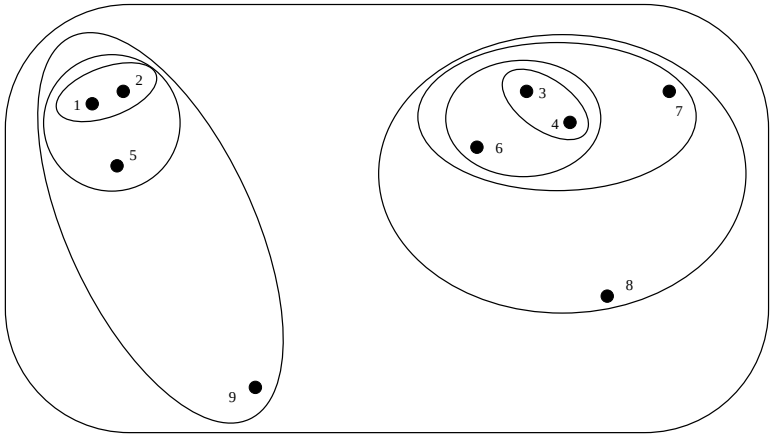

FIGURE 1. Hierarchical clustering of nine points. Ellipses around sets of points illustrate the hierarchy of containment. Initially, points 1 and 2 are closest, so they are merged; then points 3 and 4, and so on. Suppose we wanted to extract three clusters. Then, from this hierarchical clustering we would find the groupings $\{1, 2, 5\}, \{3, 4, 6, 7, 8\}, \{9\}$, since these are the last three sets to remain in the order of merging.

will focus on describing some clustering algorithms that give few strong guarantees about their output, but which have been observed to work well and efficiently in practice.

3. Hierarchical Clustering

We begin our discussion of clustering algorithms with a simple to describe method [**7**]. The idea of hierarchical (also known as agglomerative) clustering is to begin with each point from the input as a separate cluster. We then build clusters by merging clusters that are close to each other: repeatedly merge the two clusters that are closest to each other out of all pairs. This gives a hierarchy of containment, as each point in the input belongs to a succession of larger clusters. If we keep merging, we end up with a single cluster that contains all points, and the structure of the hierarchy can be represented as a (binary) tree. From this tree we can extract a set of k clusters by, for example, stopping the merging when only k clusters remain, or when the closest pair of clusters are at a distance exceeding some threshold. An example hierarchical clustering is illustrated in Figure 1.

The crucial part of this algorithm is to define the distance between two clusters of multiple points. Several natural definitions suggest themselves: it could be the smallest distance between a point in one cluster and a point in another; the greatest distance between such points; or the average distance. Each definition has its own advantages and disadvantages. For example, taking the minimum cross-cluster distance (also known as *single-link* clustering) can lead to building clusters that are "snakes": long and thin clusters, where each point is close to its closest neighbor, but the whole cluster spreads out very far. Taking the maximum distance (*complete link* clustering) favors clusters that are circular in preference to other shapes that

might better capture the nature of the data. And computing the average (*average link* clustering) can considerably slow down the computation of the clustering, which can already be somewhat slow. This is seen as one of the principal disadvantages of the hierarchical approach: even on fairly small data sets, the running time can be quite significant. We need to maintain the distance between each cluster and every other cluster. Initially, there are $\Omega(n^2)$ such inter-cluster distances, since everyone of the n input points is in its own cluster. Every time we merge a pair of clusters, we have to update the distance between the new cluster and all other clusters. Since there are up to n merges of clusters, the total cost of this algorithm is $\Omega(n^3)$ distance computations. This cost can grow to $\Omega(n^4)$ for the average link case.

4. The k-means method

Algorithm 1 The k-means algorithm [**8**]

Require: set of input *items*, x, in Euclidean space; desired number of clusters, k.

for $1 \leq i \leq k$ **do**
  $kmeans[i] \leftarrow$ random item from data
  $centroid[i] \leftarrow 0$
  $count[i] \leftarrow 0$
repeat
  for all $x \in items$ **do**
    $mindist \leftarrow 1$
    for $1 \leq i \leq k$ **do**
      if $\|x - kmeans[i]\|_2 < \|x - kmeans[mindist]\|_2$ **then**
        $mindist \leftarrow i$
    $cluster[x] \leftarrow mindist$
    $centroid[mindist] \leftarrow centroid[mindist] + x$
    $count[mindist] \leftarrow count[mindist] + 1$
  for $1 \leq i \leq k$ **do**
    $kmeans[i] \leftarrow centroid[i]/count[i]$
    $centroid[i] \leftarrow 0$
    $count[i] \leftarrow 0$;
until no items reclassified **or** repetition count exceeded
each $x \in items$ is now classified by $cluster[x]$

The k-means algorithm [**8**] is very widely used to produce clusterings of data, due to its simplicity and speed. The idea is based around clustering items using *centroids*. These are points in the metric space that define the clusters. Each centroid defines a single cluster, and each point from the data is associated with the cluster defined by its closest centroid (ties being broken arbitrarily as usual). The algorithm proceeds in rounds: in each round, every input point is inspected and compared to the k centroid points to find which is closest. At the end of every round, we compute a new set of centroids based on the points in each cluster. For each cluster, we compute the centroid of that cluster, as the "center of mass" of the points. The center of mass can be found efficiently by finding the mean value of each co-ordinate. This leads to an efficient algorithm to compute the new centroids with a single scan of the data: for each of the k clusters, compute the sum of each

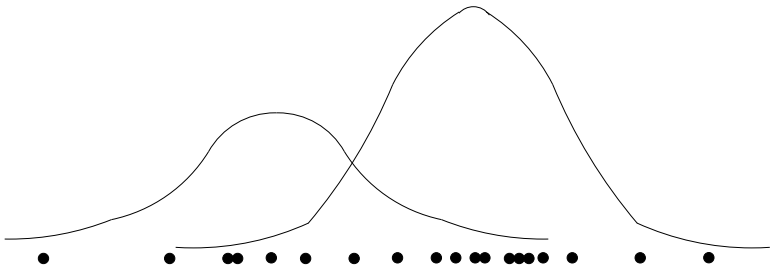

FIGURE 2. Expectation Maximization seeks to find the mixture of distributions that best explains the input set of points.

co-ordinate value of all points that are associated with that cluster, and the count of the number of points in the cluster. The new centroids can then be easily computed after all points have been allocated to clusters. The process terminates either when the clusters do not change (no points are placed in different clusters), or after a set number of iterations. The algorithm is given in pseudocode in Algorithm 1.

From this description, two factors emerge. Firstly, the process relies on the data being numerical in all attributes. The distances measured are implicitly Euclidean distance. So, in the presences of non-numeric data, then k-means cannot be applied unless some pre-processing of the data is done to convert it to a numerical format. Secondly, the results depend on the choice of the initial setting of the cluster centroids. Typically these are chosen as random points from the input, but other heuristics can be applied. It is observed that the algorithm can be very sensitive to the initial centroids. Different choices can give very different clusterings. Certain bad cases can occur: sometimes, if two centroids are very close to one another, then one tends to dominate the other, so the number of points in one shrinks to almost zero, effectively "wasting" a centroid. Also, outlier points can distort the results, either by stretching the shape of the clusters, or by taking a centroid away from the bulk of the data. Various heuristics can be applied to avoid bad cases, such as repeating the clustering a few times and picking the one with the best score, based on the k-center or k-median criterion.

A more general complaint is that the method requires k to be specified up front, and typically it is not clear in advance how many clusters there are within the data. This applies to several clustering methods, not just k-means. The response of practitioners seems to be to try various values of k until an appropriate value is found. Here at least the speed of the k-means method is an advantage: each round requires each input point to be compared to the current k centroids, and so it can completed in time $O(kn)$. Typically, a relatively small number of rounds are required before a good clustering is found.

5. Expectation Maximization (EM)

In some ways, the Expectation Maximization (EM) [**2**] approach to clustering can be seen as an extension of k-means, with a more solid theoretical underpinning. What is the model that k-means is applying to the data? It is that the each data point belongs to one of k clusters that are defined by a set of k points. The division of space induced by these points can be represented by a Voronoi diagram [**14**]. EM relaxes the assumption that every point comes from a single cluster, and instead

models the data as the result of some generative process. For example, typically EM uses a model that says that the data is being generated by a mixture of Gaussian (Normal) distributions. Each distribution gives a probability density over the whole of the space for generating points. If there are k such distributions, then the probability density function comes from taking the scaled union of these individual densities. Each of the distributions can have different parameters—in the case of Gaussians, these need only be the mean and standard deviation for one dimension; for higher dimensions, then there are more parameters to describe the shape of the distribution. This is illustrated in Figure 2. If we accept this model for the data, then the clustering process can be thought of as a search for the parameters of the generating distributions. The Expectation Maximization stage is, given the model and the data, to find the settings of the parameters of the model that best explain the data. That is, they are the most likely settings of the parameters given the data. The result of this means that we do not allocate points to clusters but rather for each data point we can evaluate the produced model at that point and see the relative probabilities that this point came from each of the k different distributions. It is this model which represents the clustering, and which can be used to predict future outcomes.

In order to generate the maximum likelihood settings of the parameters, various algorithms can be employed which, at a high level, resemble k-means. From an initial guess of the settings of the parameters, successive passes over the data refine these guess and improve the fit of the data to the current model. The details depend on the distributions used in the model (Gaussian, Log-Normal, Poisson, Discrete). For a model with a single Gaussian distribution, the sample mean is the maximum likelihood estimator. For two or more Gaussians, one can write out the expression for the mixture of these distributions, and, based on the current estimates of the parameters, compute the likelihood that each input point was generated by each of the distributions. Based on these likelihoods, we can create new settings of the parameters, and iterate. Each step increases the likelihood of the observed data given the current parameters, until a maximum is reached. Note that this maximum may be a local maximum, rather than the global maximum. The maximum that is reached depends on the initial setting of parameters. Hence we see the connection to k-means, the principal differences being the greater emphasis on an underlying model, and the way that each point has a probability or likelihood of belonging to each cluster, rather than a unique parent cluster.

A further advantage of EM is that non-numerical data can more easily be fitted into the models: for categorical data, for example, we can have a discrete probability distribution giving the probability of being in each category for each point. However, the additional cost of evaluating the model and computing the new likelihoods means that it can be slower than k-means. It also requires the user to provide a global hypothesis in advance on the model: not only do they have to give k, the number of distributions, but also describe these: are they k Gaussians, or j Poisson distributions and $k - j$ Gaussians, etc. Compared to the crispness of other so-called "hard clustering" methods, the "fuzzy clustering" produced by EM can disquiet some users.

6. Conclusion

Clustering remains a popular method for extracting hypotheses from large amounts of data. One particular advantage is that, unlike some other data mining methods, it does not require any of the input data to be "labeled", that is, inspected by an expert and tagged with a prognostication. Instead, clustering merely tries to identify groups of similar items within the data and report these back to the user. This falls into the class of "unsupervised learning" techniques, in contrast to "supervised learning", which requires a training set of data to be made available which is tagged with the appropriate class identifier. Understanding the mechanism of the clustering method is important for the user, so that they may evaluate the significance and meaning of the results of clustering. We have only discussed a few of the clustering methods that have been proposed, and mentioned a few of the factors in their use. There have been many variations and alternative methods defined in the database literature on clustering: methods such as CLARANS [**12**], DBSCAN [**3**], CURE [**4**], BIRCH [**17**], and many others.

Many software packages are commercially available that implement such clustering methods. For example, Mathematica [**9**] and Matlab [**10**] both contain routines to perform various clustering operations on input data. XLMiner is a plug-in for Excel that implements k-means clustering and hierarchical clustering [**16**]. Clustan (`http://www.clustan.com/`) is a software package devoted to cluster analysis. In addition to these and many other commercial solutions, one can also find free implementations of a variety of languages: Fortran, C++ and Java being the most popular. For more details on clustering approaches, see one of the several good quality textbooks on data mining [**6, 5, 1**], or tutorials available on the web [**15, 11, 13**].

References

[1] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.
[2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
[3] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page 226, 1996.
[4] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.
[5] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.
[6] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.
[7] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.
[8] J. B. MacQueen. Some method for the classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Structures*, pages 281–297, 1967.
[9] Mathematica. `http://www.wolfram.com/`.
[10] Matlab. `http://www.mathworks.com/`.
[11] A. Moore. k-means and hierarchical clustering. `http://www-2.cs.cmu.edu/~awm/tutorials/kmeans.html`.
[12] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–155. Morgan Kaufmann, 1994.
[13] A tutorial on clustering algorithms. `http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/`.
[14] J-R Sack and J. Urrutia, editors. *Handbook on Computational Geometry*. Elsevier Science, 1998.

[15] J. Ullman. Lecture notes on clustering. http://www-db.stanford.edu/~ullman/mining/cluster1.pdf.
[16] XLMiner. http://www.xlminer.net/.
[17] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.

Rutgers University
E-mail address: graham@dimacs.rutgers.edu