COMP5318 Knowledge Management & Data Mining Assignment 1

Enoch Lau
SID 200415765

7 May 2007

Abstract

Clustering is a fundamental task in data mining that aims to place similar data values into the same category, so that the values can be viewed as being related in some way. A common problem in clustering is automatically determining the number of clusters to be used. We examine the use of the Bayesian Information Criterion (BIC) in conjunction with an agglomerative clustering algorithm. Experiments were conducted into the efficacy of the approach.

Contents

1 Introduction

There are various approaches to hierarchical clustering, including the agglomerative and divisive approaches [2]. In this report, we use the agglomerative clustering algorithm with complete (maximum) link used as the proximity measure. After executing these clustering algorithms, however, there is still the problem of determining how the final set of clusters should be chosen from the hierarchy of clusters. A proposal is the Bayesian Information Criterion (BIC) [1], which is a model selection criterion. There are several implementation issues that are also discussed, as well as instructions for running the code. Finally, experimental results are presented, where we present the result from the algorithm; we also investigate the effect on the efficacy of the algorithm when the density and variance of the clusters are varied, and the number of dimensions is changed. Scalability is measured from an experimental point of view.

2 Agglomerative Hierarchical Clustering Algorithms

The agglomerative clustering algorithm used progressively merges clusters together, using some measure of proximity. There are various ways in which the proximity of two clusters can be measured; using single link (MIN), complete link (MAX) and average are just some approaches possible [2]. In this case, we chose the complete link (MAX) measure, where the distance between two clusters is measured as the largest distance between any point in the first cluster and any point in the second cluster [2].

The algorithm can be described as follows. Initially, each value in the data set is assigned to its own cluster. The aim is to progressively merge pairs of these clusters one-by-one, until there is only one cluster remaining. At every stage, the pair of clusters chosen to be merged is the pair with the smallest distance between them, according to the chosen measure, which, in this case, is the complete (maximum) link measure.

After the algorithm has been run, it is possible to represent the merges using a dendogram [2]. An approach that can be used to create the final set of clusters is to examine the height of a link in the dendogram with the height of the links below that; if the gap is large, then it can be said that the clusters merged were relatively far apart, and thus should be kept separate. Another approach, as proposed in [1], is to select the clustering that optimises the Bayesian Information Criterion (BIC).

3 Clustering with the Bayesian Information Criterion

As noted in [1], the ideal situation is where the number of clusters is chosen automatically according to the complexity of the data set, with more clusters being needed for data that is more complex. The authors propose the use of the Bayesian Information Criterion (BIC), which is a model selection criterion in the statistics literature, as a measure of the worth of a particular clustering, $\mathscr{C}_n$. BIC is penalised according to the number of parameters in the model, which avoids the problem of overfitting the data. More precisely, if we wish to cluster a data set $\mathscr{X} = \{x_i \in \mathbb{R}^d : i = 1, \ldots, N\}$, and $\mathscr{C}_k = \{c_i : i = 1, \ldots, k\}$ is a clustering with k clusters, with n_i being the number of samples in c_i, then:

$$\begin{aligned} BIC(C_k) &= \sum_{i=1}^{k} \left\{ -\frac{1}{2} n_i \log|\Sigma_i| \right\} \\ &\quad - Nk\left(d + \frac{1}{2}d(d+1)\right) \end{aligned} \tag{1}$$

Here, each cluster c_i is a multivariate normal distribution $N(\mu_i, \Sigma_i)$, where μ_i and Σ_i can be estimated from the sample data.

Although the clustering with the highest BIC value is desired, it is intractable to search for the maximum via a global search and a greedy approach is offered. From Equation (1), the increase in the BIC value by merging two nodes s_1 and s_2 into s, with n being the number of samples in s, is:

$$\begin{aligned} &-\frac{1}{2} n \log|\Sigma| + \frac{1}{2} n_1 \log|\Sigma_1| + \\ &\frac{1}{2} n_2 \log|\Sigma_2| + N\left(d + \frac{1}{2}d(d+1)\right) \end{aligned} \tag{2}$$

As suggested in the paper, we terminate the above agglomerative hierarchical clustering algorithm when the BIC is negative.

4 Implementation Details

The agglomerative hierarchical clustering algorithm, with and without the BIC, is implemented in MATLAB. To invoke the code, all the files must be placed into the same directory, and MATLAB must be opened to the directory in which they reside.

4.1 Agglomerative Clustering Algorithm

The clustering algorithm (without the BIC) is implemented in `basiccluster.m`, as a function `basiccluster`.

Information about the clusters are stored in a matrix, with each row corresponding to each cluster. The first column is the cluster id number, used for generating the list of operations; the second column is the number of samples in the cluster, say n_i. In the same row, the following n_i columns indicates the samples that are contained in the cluster; they are indexed by their row in the input matrix. The proximity of the clusters is stored as a proximity matrix, where the (i, j)-th entry (for $i > j$) is the distance between clusters i and j according to the complete link measure; note that only the values below the diagonal are used.

4.2 Bayesian Information Criterion

The clustering algorithm, with the BIC integrated, is implemented in `biccluster.m`, as a function `biccluster`. The BIC increase formula from [1] is contained in `bicincrease.m`, as a function `bicincrease`.

The interesting thing of note is the handling of clusters of size 1, which occur at the start of the algorithm, when all data points are placed into its own cluster of size one. The problem here is that the covariance matrix for a cluster of size one is the scalar 0, and Equation (2) requires that we take the logarithm of the determinant of the covariance matrix; the logarithm of 0 is undefined. To allow clusters to form, the code does not use Equation (2) if both clusters are of size one, which is a good indication that we are at the beginning of the merging phase.

However, the BIC increase formula in Equation (2) was found to perform exceptionally poorly, and for the experimental results, they are repeated using Equation (2) and a modified version of the equation for the increase in BIC value, which was found to perform better:

$$\begin{aligned}&-n\log|\Sigma| + n_1\log|\Sigma_1| + \\ &n_2\log|\Sigma_2| + (N-n)\left(d + \frac{1}{2}d(d+1)\right)\end{aligned} \tag{3}$$

The $(N - n)$ modification is at least motivated by the fact that the last term, $N\left(d + \frac{1}{2}d(d+1)\right)$ is too large for large d or large n, and the BIC increase is almost always positive, leading to all the points being placed into the same cluster.

4.3 Invocation

The `basiccluster` function takes one parameter, which is a matrix where the number of columns is equal to the number of dimensions of the data, and each row corresponds to a particular data sample. This function returns a matrix where each row consists of two values, namely the clusters that were merged at a particular step, in the same vein as the MATLAB Statistics Toolbox function `linkage`. A sample of how to invoke it is included in `basicclustertest.m`.

The `biccluster` function takes one parameter, as per the `basiccluster` function. It returns the matrix of operations as before, but it also returns the number of clusters created, and the assignments of the sample values to clusters. The experiments as outlined below are contained in `bicclustertest.m`.

In addition, the `drawplot.m` file draws Figure (2) and `scalability.m` file draws Figure (6). These scripts can be run directly from MATLAB by typing the script's name on the command line.

Documentation for each of the functions can be obtained by typing `help functionname`.

5 Experimental Results

As mentioned previously, the experiments were repeated using the (corrected) equation from the paper (Equation (2)), and the modified equation (Equation (3)). Each experiment was repeated 10 times, because the data placed into the algorithm is generated randomly, and the average is reported in the tables. The data placed into the algorithm is a mixture of one or more normal distributions.

5.1 First Experiments

No. of clusters	Using (2)	Using (3)
1	1.0	1.6
3	1.0	2.5
6	1.0	2.8

Figure 1: Changing the number of clusters: each two-dimensional cluster has a standard deviation of 1, and is at least 100 units away from another cluster

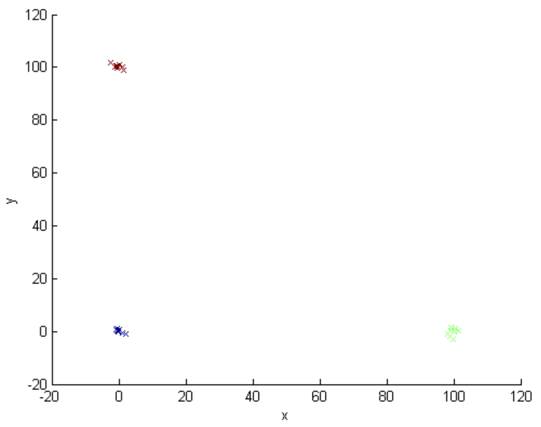

Figure 2: Three clusters, each a multivariate normal distribution of variance 1, clustered using the algorithm with Equation (3)

The first experiments simply modified the number of clusters to see how well the algorithm detects the number of clusters. In Figure (1), we used two-dimensional data, and changed the number of clusters. The algorithm using Equation (2) performs poorly, while the modified equation works better. Even with the modified equation, for a large number of clusters, the algorithm underestimates the number of clusters. In Figure (2), we can see the three clusters (the same data as the second row of Figure (1)) being clustered into three distinct clusters.

5.2 Density of Clusters

Rel. density	Using (2)	Using (3)
1	1.0	2.0
3	1.0	2.7
6	1.0	2.7

Figure 3: Changing the density of one of the clusters: the relative density of one of the clusters is changed by adding more points into that cluster, while the other clusters remain the same; the expected number of clusters is 3

We changed the density of one of the clusters to see whether the algorithm is adversely affected by varying densities; the results are in Figure (3). When the density of one of the clusters was increased, it appears that the algorithm is better able to predict the number of clusters (on average, detecting 2.7 clusters when there are 3 in actuality). Increasing the density further appeared to have made no difference.

5.3 Variance of Clusters

Variance	Using (2)	Using (3)
1	1.0	2.4
3	1.0	3.0
6	1.0	3.0

Figure 4: Changing the variance of one of the clusters: the variance of one of the clusters is changed, while the other clusters remains at 1; the expected number of clusters is 3

We changed the variance of one of the clusters to see if variance affects the ability of the algorithm to detect clusters; the results are in Figure (4). It would appear that the algorithm performs better, detecting 3 clusters in all cases, when the variance of one of the clusters is changed to 3, and then 6.

5.4 Number of Dimensions

Dimensions	Using (2)	Using (3)
1	3.0	6.0
2	1.0	3.0
3	1.0	1.7
4	1.0	1.0

Figure 5: Changing the number of dimensions: the number of dimensions for the data points; the expected number of clusters is 6

Finally, we changed the number of dimensions to see the affect of dimensionality on the data; the results are reported in Figure (5). Surprisingly, the algorithm using Equation (2) produced a non-one result for the one-dimensional case. Similarly, the algorithm using Equation (3) performs better with lower-dimensional data. Clearly, as the number of dimensions increases, the ability of the algorithm to calculate the number of clusters decreases dramatically.

5.5 Scalability

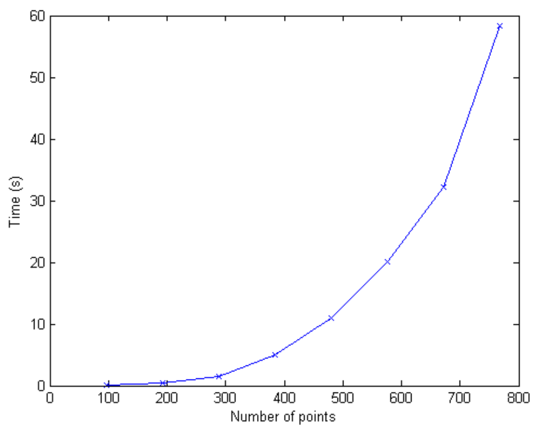

Figure 6: The scalability of the algorithm, determined experimentally

Finally, the scalability of the algorithm was examined experimentally by increasing the number of points in each of the clusters, and measuring the time taken for the algorithm to complete. The times were measured on a Pentium M 1.6GHz notebook computer with 1.25GB RAM and MATLAB 7.0 with no other open applications. Clearly, the algorithm does not have linear complexity, and appears to be of the order of $O(n^k)$ for some low k.

6 Conclusion

We described an agglomerative hierarchical clustering algorithm and the Bayesian Information Criterion, as proposed in [1]. However, through experiments, we found that the algorithm does not perform adequately using the equation as described in the paper, but by tweaking the equation, the algorithm could be made to perform better. We described some of the implementation issues, including internal representation of clusters, and described the files used for this report. Finally, experimental results were presented, which showed that the algorithm, with the modified BIC equation, performs best on low-dimensional data, and has the ability to withstand changes in variance and density. The scalability of the algorithm was measured experimentally, and was clearly demonstrated to be greater than linear.

References

[1] S. Chen and P. Gopalakrishnan. Clustering via the Bayesian Information Criterion with Applications in Speech Recognition. *Proc. ICASSP*, 2:645–648, 1998.

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