Iterative Reclassification in Agglomerative Clustering

Nicholas A. HEARD*

In model-based clustering of complex data a probability model, typically a finite mixture probability model, forms the basis of the distance measure between any pair of clusters. The idea of model-based clustering was popularized by the framework and accompanying software of Fraley and Raftery (2002). In particular, model-based agglomerative hierarchical clustering is now a frequently used approach for probabilistic grouping of data, due to the speed and simplicity of implementation. This paper investigates deficiencies in the clusterings proposed from this popular approach, and presents a review of small refinements and extensions to the procedure with differing performance gains and computational costs. The improvements are illustrated through application to simulated and real data examples, including the clustering of gene expression time profiles. Some of the proposed improvements to agglomerative clustering are, like the procedure itself in its usual form, deterministic; perhaps surprisingly though, the best overall results here are obtained via a stochasticized version of the entire procedure. Whilst the focus of this paper is probability model-based clustering, many of the schemes presented are equally applicable to agglomerative clustering under any distance measure.

The simulated data from this paper along with the C++ code used for implementing the algorithms for all of the examples can be obtained online from the Supplemental Material.

KEY WORDS: Model-based clustering; Agglomerative clustering; Iterative reclassification; Stochastic search.

1. INTRODUCTION

Due to its speed and simplicity of implementation, agglomerative clustering is widely regarded as the default method in applied science for splitting a set of data into homogeneous groups. Here,

*Nicholas A. Heard is Lecturer in Statistics, Department of Mathematics, South Kensington Campus, Imperial College London, SW7 2RH (Email: *n.heard@imperial.ac.uk*).

for a collection of n data points $\{y_i \in \mathbb{R}^p : i = 1, \ldots, n\}$, this paper examines some simple extensions of model-based agglomerative clustering for finding improved locally optimal partitions. Importantly, these extensions will retain much of the simplicity of the basic agglomerative algorithm.

Agglomerative clustering (Johnson, 1967) is an iterative clustering strategy which begins with each datum y_i in its own cluster and successively merges cluster pairs chosen to maximize an objective function. The term "model-based" refers to the case where a probability model $\pi(\mathfrak{C})$ acts as the clustering objective function measuring the quality of rival partitions $\mathfrak{C}$.

The performance of the model-based agglomerative clustering algorithm, which is fully defined in Section 2, in finding a good partition will depend both on the structure of the assumed probability model and the nature of the data; in particular, the number of data points n and the dimension p of each datum y_i can be important.

In practice it can be observed that model-based agglomerative clustering will often be biased towards building up one large cluster before moving onto the next. Sections 3 and 4 of this paper present schemes which seek to negate this bias and build up clusters in a more balanced way, with the aim of obtaining higher scores for the objective function π. When these schemes are applied to several simulated and real data clustering problems in Section 5, the best performance of all is achieved by a stochastic variety of agglomerative clustering where the choice of clusters to merge is made at random.

2. AGGLOMERATIVE HIERARCHICAL CLUSTERING

A fully probabilistic approach to cluster analysis such as that of Bayesian inference would formally require full exploration of a probability model π over the entire partition space for accurate inference to be performed. However, for n data points the number of possible partitions is given by the n^{th} Bell number, which for the smallest example here of 82 points corresponds to an order of 10^{89} possible models, and so a full exploration of the space is not possible. Instead, stochastic search methods such as reversible jump MCMC (Richardson and Green, 1997) with elaborate transition kernels (Jasra et al., 2007) can be used in these circumstances to explore a small but potentially high probability subset of this discrete space.

Hierarchical clustering methods provide a more straightforward, deterministic approach to the search for a high probability clustering. Agglomerative hierarchical clustering is very popular

for two reasons. Firstly, the agglomerative algorithm is very quick, evaluating just $(n-1)^2+1$ highly related partitions for n data points in the contexts presented here. Secondly, it provides a much richer output than flat clustering techniques, such as the *k-means* procedure (MacQueen, 1967); a binary tree $\mathfrak{T}$ of nested partitions is obtained from the hierarchy of models visited, of which the proposed optimal clustering $\mathfrak{C}$ is just one cut. The number of data groupings to report is often unclear, particularly so in distance-based clustering, and the tree suggests a range of possible clusterings at different levels of granularity. Diagrammatically, the hierarchy is best presented as a dendrogram indicating the order in which the agglomerations took place (see Figures 1 and 2).

2.1 AGGLOMERATIVE CLUSTERING ALGORITHM

Model-based agglomerative clustering is routinely applied in many applied science contexts, including biology, medicine and psychology. Motivating this paper is the frequent use of agglomerative clustering in the biologically interesting context of clustering of gene expression; similarity measure-based examples can be found in the regularly cited papers of Eisen et al. (1998) and Spellman et al. (1998), and model-based examples appear in Yeung et al. (2001), Ramoni et al. (2002) and Heard et al. (2006).

The basic model-based agglomerative clustering algorithm is fully written out in Algorithm 1 in fairly general terms so that later modifications can be easily explained. In words, each data point starts off in its own singleton cluster and then at successive iterations two clusters are merged so that the resulting change in the objective function π is optimized; this is repeated until all of the data reside in a single cluster. The best partition visited is then reported as the optimal clustering, with the objective score a measure of partition quality which can be compared for different algorithms. Additionally, the average objective score of models visited within the dendrogram similarly provides a measure of quality of the full tree hierarchy, which may also be reported. This agglomerative clustering algorithm will be referred to later simply as AC.

For efficient implementation of Algorithm 1, the upper triangle of the symmetric cluster similarity matrix S, which at its largest is of dimension $n \times n$, is always stored; then following a merger only the $(k-1)$ affected entries of S are updated (lines 18-22), with its dimension also reducing by one. Furthermore, various sufficient statistics can be stored for each cluster, according to the precise nature of the probability model.

Important early methodological citations for model-based agglomerative clustering are Ward

(1963) (in the more general context of π being some *objective function* to maximize, not necessarily a probability model) and Fraley and Raftery (2002). Motivated by building better tree hierarchies rather than just optimal output partitions, Heller and Ghahramani (2005) extended model-based hierarchical clustering to use a recursive objective score for clusters made up of a mixture of the usual model probability for a cluster combined with the product of the objective scores of the first two sub-clusters in the preceding hierarchy. This method, referred to as Bayesian Hierarchical Clustering (BHC), has an interpretation as an approximate inference model for Dirichlet Process mixtures (Escobar, 1994), and is implemented later on in this paper in the numerical comparisons.

Related to these methods are other approaches which place a probability distribution on the entire binary tree hierarchy rather than on a particular cut (see, for example, Teh et al., 2008). This is a much more computationally challenging framework and so is not considered further here, although there is scope for generalizing the methods presented here to that setting.

2.2 Limitations of Agglomerative Clustering

Whether being used to provide a final clustering solution or simply to give a starting model for more complicated search procedures, the aim of agglomerative clustering remains the same: to find 'good' partitions of the data with respect to the objective function. However, the potential quality of these partitions is limited by the algorithm.

First, it should be noted that in agglomerative clustering mergers are irreversible, so that once two clusters are chosen to merge, their elements will remain together throughout the remainder of the hierarchy. So in particular there will be data points which have merged into clusters early on in the algorithm which later on, when other "rival" clusters have been formed, might have been better placed elsewhere.

Second, if one observes Algorithm 1 in practice, it commonly happens that one large cluster quickly starts to form with the rest of the data points remaining as singleton clusters. Examples of this can be seen in the dendrograms in Figures 1 and 2 later on. This phenomenon seems particularly prone to occurring in Bayesian probability model-based clustering when informative priors are being used. The large, early clusters that form typically contain data points which are somewhat outlying relative to the prior specification and so marginally have low predictive probability, but actually can be separated under the same model using alternative search techniques.

Some of these issues can be alleviated by selecting different prior parameters for a Bayesian

probability model. However, this would go against the Bayesian paradigm, where all proper subjective prior distributions should be admissible. It is undesirable to constrain the range of allowable parameter choices of a probability model because of contrasting difficulty of inference.

The objective of this paper is to present simple varieties of the agglomerative clustering Algorithm 1 which can be easily deployed to avoid the hierarchy from deviating towards paths containing such highly sub-optimal partitions and to identify the circumstances in which each method is appropriate. Some varieties will have comparable run-times with Algorithm 1 and none are prohibitively slow. So the intention here is to suggest complementing the use of Algorithm 1 by additionally running a selection of other suitable methods in parallel. The best clustering found across these methods and its surrounding hierarchy can be reported.

3. ITERATIVE RECLASSIFICATION

The variations to AC Algorithm 1 considered here look to reclassify data points to different clusters once mergers have taken place. For an output partition $\mathfrak{C}$, it is desirable that any shift of a data point y_i to a different cluster in $\mathfrak{C}$, or into a singleton cluster, should cause a decrease in the objective function π. In practice, after performing agglomerative clustering Algorithm 1 on moderately large data sets a substantial proportion of data points are not in their optimal cluster, so when reclassified as belonging to a different cluster cause an increase in π. For example, for the Anopheles gene expression data in Section 5.3 with 2,771 genes to cluster, 377 are initially sub-optimally classified in this sense after applying Algorithm 1.

To address this issue, a reclassification step is introduced to reallocate subjects to their optimal clusters. The following reclassification algorithm can be embedded within the agglomerative clustering Algorithm 1 to improve the final output partition from this method. The reclassification can simply take place once at the end of agglomerative clustering immediately after line 24 of Algorithm 1, a strategy to be known as ACIR; or reclassification can be performed after each merger in the agglomerative scheme so immediately after line 15 of Algorithm 1, a strategy to be known as AIRC.

3.1 ITERATIVE RECLASSIFICATION ALGORITHM

Algorithm 2 gives details of a simple sequential reclassification algorithm for a given partition $\mathfrak{C}$. In words, at each iteration of this algorithm the subject whose reallocation to a different, optimal cluster will cause the biggest increase in π is selected to move. This is repeated until no subjects are suboptimally classified. For efficient implementation of Algorithm 2, an additional $n \times k$ subject-cluster similarity matrix R is stored; then following any reclassifications (or mergers if applied during, say, AC) only the affected columns of R are updated (lines 6-10). With these efficiencies the algorithm has a level of complexity which is $O(n)$ at each iteration.

A useful property of Algorithm 2 is that convergence is guaranteed; every iteration causes an increase in π, and the search space is finite. In contrast, a batch reallocation strategy where all misallocated subjects are reallocated at each iteration has no such guarantees, as single iterations of the algorithm can cause π to decrease. This lack of convergence can be seen in practice, as the algorithm can often find itself cycling indefinitely between a small number of partitions. Computationally there is little difference between sequential and batch reclassification, so the latter holds no real advantages.

3.2 PAIRWISE RECLASSIFICATIONS

It is a natural extension to consider reclassification moves involving subsets of data points, or at least pairwise shifts of data points, with perhaps say one subject joining a cluster whilst another moves out. For example, there can be a slight outlier in a cluster which has no preferred alternative cluster, that once forced to move out will allow other, better fitting subjects to then join this cluster.

Suppose an iterative reclassification clustering algorithm has been performed until convergence, so that no possible subject move directly increases the objective function. It is then proposed that pairwise moves can be considered. To implement this in practice, in turn each data subject i is proposed to move from its existing cluster $c = c(i)$ into a new singleton cluster (as is possible in Algorithm 2, and note that at this stage there is no preferable move to an existing cluster). But then rather than accepting or rejecting based on change in objective score, the shift is temporarily accepted. Then, Algorithm 2 is run, concentrating on proposals to move more subjects out of the affected cluster c to any other cluster, or to move the any of the remaining subjects into the affected cluster c. If no overall gain in objective score is yielded once convergence is reached, all of the

changes are undone. This is repeated across subjects until a move is made permanent, and the entire process can be repeated until convergence. Computationally, this is clearly a more demanding strategy, adding a level of complexity which is $O(n^2)$ at every iteration.

Searching for pairwise shifts after the convergence of strategy ACIR or AIRC is to be referred to as ACIR2 and AIRC2 respectively.

3.3 Rebuilding the Dendrogram

As mentioned in Section 2, in many contexts an important output of agglomerative clustering is the hierarchy itself. On first glance, reclassification strategies like ACIR and AIRC might appear to carry the disadvantage of breaking this hierarchical structure. However, any agglomerative clustering scheme involving reclassification can be followed with a run of a constrained version of AC, Algorithm 3, forced to pass through the optimized clustering and create a new hierarchy. That is, more generally the agglomerative clustering Algorithm 1 can be constrained to pass through an arbitrary partition $\mathfrak{C}^*$ by Algorithm 3, which at lower levels of the hierarchy simply disallows the merger of two clusters if their union is not a subset of a cluster in $\mathfrak{C}^*$.

Of course, running Algorithm 3 may lead to finding a new optimal partition again with a different number of clusters, in which case the reclassification procedure can be repeated if so desired; at each stage until final convergence, only improved partitions according to π are being discovered.

4. Stochastic Agglomerative Clustering

A stochastic version of agglomerative clustering is now considered. Stochastic optimization methods, such as simulated annealing or shotgun stochastic search (Hans et al., 2007) are often favored in complex high or variable dimensional settings to avoid local maxima and provide a wider search of the model space. A natural implementation of stochastic agglomerative clustering is given in Algorithm 4.

In stochastic agglomerative clustering, a cluster is chosen for merger with probability proportional to the gain in objective function π which will be caused by merging with its nearest neighbor. So the agglomeration causing the highest increase in π goes from being the automatic next merger to simply being the most likely one. Without further refinement, this would be unlikely to outper-

form the usual deterministic agglomerative algorithm, as too many suboptimal mergers are being purposefully introduced. However, when combined with the reclassification methods of Section 3, stochastic agglomerative clustering becomes a very competitive method; in fact, through the numerical analysis of several data sets presented in the next section a stochastic method will be seen to be the most robustly performing algorithm considered in this paper. Some explanations for this high level of performance are given in the discussion at the end of the paper.

Notationally, stochastic agglomerative clustering (SAC) with iterative reclassification at the end will be referred to as SACIR, whilst SAC with iterative reclassification at each merger will be referred to as SAIRC. Following these two strategies with pairwise reclassifications will be referred to as SACIR2 and SAIRC2 respectively.

5. EXAMPLES

The following methods are now compared for performance over a variety of data sets: AC of section 2; ACIR, ARIC, ACIR2 and AIRC2 of section 3; SACIR, SAIRC, SACIR2 and SAIRC2 of section 4; the BHC method of Heller and Ghahramani (2005) and also BHC followed by flat model-based iterative reclassification (BHCIR) and then, for the first example, pairwise iterative reclassification (BHCIR2). To implement BHC, following Heller and Ghahramani (2005) the free parameter of their approximate Dirichlet Process formulation is chosen to maximize the root node posterior probability; whilst this leads to an increase in computation time which will not be included in the results presented here, the expense is worthwhile as the extra effort causes the BHC schemes to be very competitive for some data sets.

The data sets considered are: a simulated data set, generated from a mixture of four bivariate normals, so as to match the simulations described in Heller and Ghahramani (2005); the *Galaxy*, *Acidity* and *Enzyme* data sets used for assessing Bayesian methods for mixtures of normals in Richardson and Green (1997); and gene expression time profile data from Dimopoulos et al. (2002), modeled as a piecewise linear regression clustering model as in Heard et al. (2006).

Four criteria will be used to assess the performance of each algorithm. Firstly, the maximal (log) model probability found by the algorithm; secondly, the (log) average model probability across the output hierarchy; thirdly, the (log) probability of the data given the entire hierarchy, using the formula of Heller and Ghahramani (2005); fourthly, in the case of simulated data where true class memberships are known, the *purity* of the hierarchy (Heller and Ghahramani, 2005). To

define purity, for any two subjects with the same class label let the purity of their clustering within the hierarchy be the proportion of data points sharing their class label in the smallest cluster that contains the two subjects; the purity of a hierarchy is the mean value of this quantity across all such pairs.

Computational run times for the algorithms are given for each data set. All computations were carried out using a single processor of a 2.13GHz Intel Core 2 Duo MacBook Air. C++ source code used for implementing all of the algorithms for all of the data sets analyzed here is available from the Supplemental Material.

5.1 Synthetic Bivariate Gaussian Data

Following Heller and Ghahramani (2005), 50 data points were simulated from each of four bivariate normal distributions. Here the normals were chosen to be spherical, had means located on the vertices of a square and had differing variances. The data obtained from simulation are shown in Figure 1, and are available as supplementary material to this paper. For a Bayesian analysis of these data, the mean and variances for the four normals were treated as unknown and following respective conjugate priors of standard normal and inverse gamma with unit shape and scale parameters.

The remaining plots in Figure 1 show the dendrogram outputs from a selection of the algorithms. The "greediness" of the dendrogram for standard agglomerative clustering (AC) is visible here, with increasingly large clusters being built up in isolation. Recalling that each true cluster should be of size 50, even a casual inspection of this dendrogram reveals that it cannot be right. The dendrogram for BHC in Figure 1 is at the other extreme, building up many small clusters before eventually uniting them into larger clusters. Finally in Figure 1, the dendrogram for AIRC represents something of a compromise between these two extremes; and encouragingly, there are obvious boundaries in the dendrogram after approximately 50, 100 and 150 data points.

Table 1 provides a quantitative comparison. Against the plain agglomerative clustering and Bayesian Hierarchical Clustering algorithms, all of the iterative reclassification strategies lead to large performance gains. Interestingly, ACIR and outperforms BHC and BHCIR outperforms AC, even on their own metrics. There is little to choose between the various IR strategies, all are performing well with purity scores above 95% which suggest the methods are able to successfully classify most points somewhere within the hierarchy.

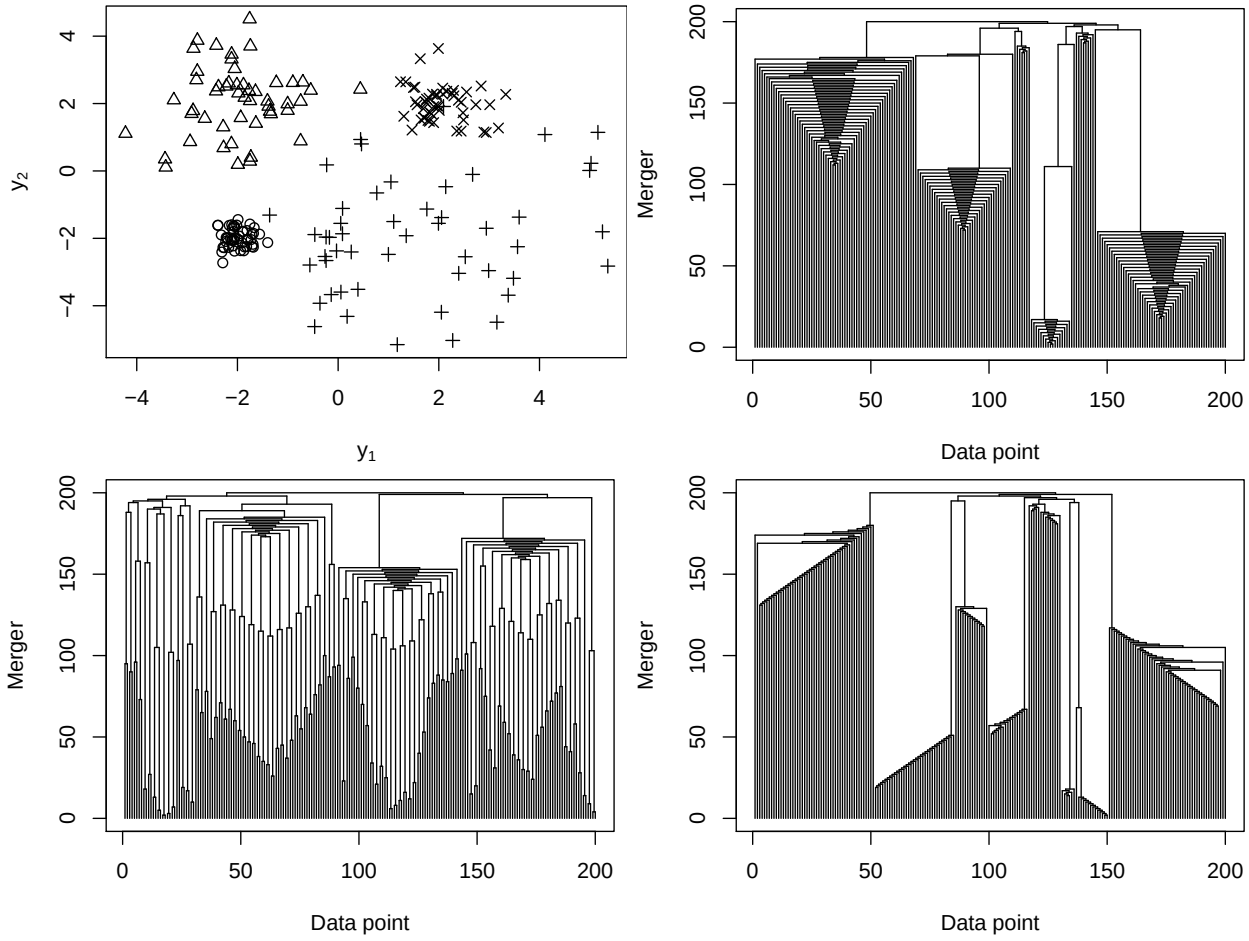

Figure 1: Synthetic data simulated from four bivariate normals (top left); dendrogram output for these data from the clustering algorithms AC (top right), BHC (bottom left) and AIRC (bottom right).

Running the additional pairwise search for reclassifications leads to very modest gains in performance in each case, particularly when compared against the gains made by the initial strategies of moving just one subject at a time. However, computation time is still sufficiently low here that there seems little to lose in extending the search in this way.

5.2 GALAXY, ACIDITY AND ENZYME DATA

Richardson and Green (1997) used three real univariate data sets to illustrate the benefits of reversible jump Metropolis-Hastings samplers, referred to there as the *Galaxy*, *Acidity* and *Enzyme* data sets. Respectively these concern distributions of galaxy velocities, lake acidity indices and enzyme activity levels in the blood; further description of these data (along with details of how to obtain them) can be found in their paper, but for the purposes of this investigation simply note that the aim in Richardson and Green (1997) was to fit these data with an unknown mixture of normals. Since mixture density estimation can be viewed as a clustering problem, these data sets provide a good benchmark for performance of the agglomerative clustering methods considered here.

Table 2 shows summary results from applying different agglomerative clustering schemes to the Galaxy, Acidity and Enzyme data, using a simple conjugate mixture of normals probability model as in Section 5.1. The three data sets have $n = 82$, $n = 155$ and $n = 245$ data points respectively, and so are presented in increasing order.

Immediately apparent from the table is that as n becomes larger, standard agglomerative clustering methods (either AC or BHC) fare increasingly badly; for the larger Enzyme data set, any of the suggested modifications to the algorithm lead to massive gains in objective function score; for the smaller Galaxy data set a less marked improvement is still achieved in all cases. The computational times are sufficiently low not to merit discussion.

To see the cause of the improved performance, one can look to the dendrograms of the agglomerative clustering schemes. Figure 2 shows the dendrograms for the Enzyme data under standard AC and the best performing method here, AIRC2. The AC dendrogram exhibits the problematic behavior of building up one particularly large cluster without interruption, resulting in discovery of low probability partitions. In contrast the rebuilt dendrogram of AIRC2 has a much more balanced structure and contains much higher probability partitions.

Whilst it is interesting to note that the optimal partitions found in each case vary considerably, it should be remembered that any questions of model choice and overfitting are irrelevant to this

study; model-based clustering is concerned solely with finding configurations with high values for the objective function π, whose structure should be considered a fixed quantity here. However, to see an illustration of the realized change in the fitted models, the predictive distributions for the Enzyme data implied by the optimal partitions proposed by some of the schemes are shown in Figure 3. At least in this case, the refined methods all explore local modes which are ignored by AC. Even between the revised methods the fitted models differ considerably, suggesting the value of these methods can come from implementing each in parallel to obtain a much richer output.

5.3 Gene Expression Data

The motivating problem for this investigation of agglomerative clustering was the grouping of gene expression time profiles from microarray experiments with the aim of generating hypotheses for gene function (see, for example, Eisen et al., 1998). In this context the data will not be univariate, but rather have dimension $p > 1$ corresponding to the number of time points at which gene activity has been measured. Typically the number of genes on the array, n, will be several thousands, and so $n >> p$.

A typical example is the *Anopheles gambiae* mosquito gene expression data of Dimopoulos et al. (2002). Using loosely the same probability model of Heard et al. (2006) based on piecewise linear regressions on the time domain with normal errors, these data are now reanalyzed using the different agglomerative clustering schemes. The data comprise $n = 2771$ genes measured after infection with a bacterial agent at $p = 6$ unequally spaced time points and the aim is to find groups of genes within the data which are approximately co-regulating over time as measured by the probability model π. The summary of results of agglomerative clustering with these gene expression data are shown in Table 3. The time taken to progress through iterations of each clustering algorithm is shown in Figure 4. In this respect, the schemes which have IR at each iteration are seen to diverge from the plain clustering schemes at an increasing rate; as agglomeration reaches completion and a smaller number of large clusters remains, there is much more reclassification work to be done.

All of the simple iterative reclassification methods again cause very substantial gains in performance, the very best coming from the stochastic methods SAIRC. Note from Table 3 that the clustering arrived upon from this method has been subjected to the greatest number of reclassifications. In contrast, the performance of BHC deteriorates here, as even with an optimized parameter

the algorithm immediately builds up a meaningless cluster containing 84% of the data points without interruption.

For this larger data set, AIRC and SAIRC are now revealed to have a much higher computation cost than ACIR and SACIR. But equally, their performance is also revealed to be far superior; too many suboptimal merges have been performed by the plain clustering methods so that iterative reclassification afterwards is insufficient, too much bad cluster structure has been established. Nonetheless, as demonstrated in Figure 5 which shows a section of the hierarchy of model probabilities obtained under these schemes, a large proportion of the available improvement in the cluster hierarchy is realized by the simple deterministic ACIR strategy, and so for huge data sets where computation becomes critical this could be important. Interestingly though, the hierarchies of AC and ACIR differ almost as much as those of ACIR and SAIRC; if the ACIR optimal clusters were considered the truth, the AC tree has a purity of 65%, whilst if SAIRC optimal clusters were true, the ACIR tree has 69% purity.

The pairwise reclassification method SAIRC2 leads to only the slightest of gains in performance from SAIRC, while AIRC2 is unable to improve on AIRC at all. Against this, the pairwise methods are causing a considerable increase in computation time.

6. DISCUSSION

This paper has demonstrated how iterative reclassification schemes which are simple to code can substantially improve the performance of commonly used agglomerative clustering algorithms such as AC or BHC. This gain in performance has been two-fold. First looking from a flat perspective, applying these simple strategies has led to discovery of cluster configurations with much higher objective score in all of the examples. At the very least this signifies tighter, truer clusters according to the assumed probability model, although the final significance of this will depend on how well the probability model has been specified; the latter is considered a fixed quantity here and not a subject for discussion in this paper, although unsatisfactory clusters obtained from these methods might well suggest that the probability model should be revised. Second, when reviewing the entire hierarchy produced by agglomerative clustering, these gains in objective score were observed in configurations throughout the dendrogram; furthermore, in the case of simulated data where the true clusters are known, substantial increases in the purity of the tree were achieved by virtue of early misallocations of subjects being rectified by the reclassification.

Extending the reclassification strategies to allow pairwise moves of subjects led to only small improvements in performance. Furthermore, the computing time required to realize these small gains is relatively high. In summary, these pairwise methods are hard to justify unless there is no urgency for results and the very best performance is sought.

As noted earlier, more sophisticated, tailored MCMC or SMC schemes would be preferred for a more thorough investigation of the partition distribution. But even then, either as provision of an initial value or as a high probability reference point for comparative assessment of convergence, the output of an 'off the shelf' algorithm requiring no tuning still has great value.

Beyond these methods, making further ground in optimization with standard split/merge/shift Metropolis-Hastings proposal samplers is unlikely. These three moves essentially duplicate the agglomerative and iterative reclassification steps in, say, ACIR, and so typically lead to no further increases in the objective function. Pairwise reclassifications can similarly be thought of as analogous to delayed rejection in MCMC (Green and Mira, 1999).

The strong performance of a stochasticized version of this deterministic algorithm is slightly unintuitive, as there is nothing inherently intelligent in randomly making sub-optimal choices. However, this randomness certainly breaks up the bias of AC towards building up big clusters before moving on. Additionally the compensation for making suboptimal moves is that by visiting this small neighborhood of locally suboptimal partitions it can become necessary that more iterative reclassifications are performed and so, importantly, more partitions are visited; and crucially, any locally suboptimal move that does not lead directly to improvement in objective score is immediately undone by reclassification in SAIRC. The tabulated results for the stochastic methods were obtained when applying a default seed value of 0 for the GNU Scientific Library random number generator. Varying this seed can lead to the same or different (better or worse) results; thus running the stochastic schemes in parallel from a multitude of seeds gives a further method for exploring a slightly increased neighborhood of models if desired.

7. SUPPLEMENTAL MATERIALS

Bivariate Gaussian mixture data Plain text file containing the bivariate data analyzed in Section 5.1, simulated from a four-component mixture of Gaussians. Each line of the file is a data point in 2-D space appended with a class label from $\{1, 2, 3, 4\}$. (synthetic_data.txt.tar.gz, GNU zipped tar file)

C++ code Directory containing the C++ source code and makefile for compiling the clustering program used in all of the examples. An example shell script for running the code on the bivariate Gaussian mixture data is included which illustrates how to select the different algorithms from the paper. Instructions for compilation and guidance for running the code are given in the included README file. (splinecluster_jcgs.tar.gz, GNU zipped tar file)

REFERENCES

Dimopoulos, G., Christophides, G. K., Meister, S., Schultz, J., White, K. P. and Barillas-Mury, C andKafatos, F. C. (2002) Genome expression analysis of Anopheles gambiae: Responses to injury, bacterial challenge and malaria infection. *Proc. Nat. Acad. Sci. USA*, **99**, 8814–8819.

Eisen, M. B., Spellman, P. T., Brown, P. O. and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Nat. Acad. Sci. USA*, **95**, 14863–14868.

Escobar, M. D. (1994) Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, **89**, 268–277.

Fraley, C. and Raftery, A. E. (2002) Model-based clustering, discriminant analysis, and density estimation. *J. Amer. Statist. Assoc.*, **97**, 611–631.

Green, P. J. and Mira, A. (1999) Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika*, **88**, 1035–1053.

Hans, C., Dobra, A. and West, M. (2007) Shotgun stochastic search in regression with many predictors. *Journal of the American Statistical Association*, **102**, 507–516.

Heard, N. A., Holmes, C. C. and Stephens, D. A. (2006) A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: An application of Bayesian hierarchical clustering of curves. *J. Amer. Statist. Assoc.*, **101**, 18–29.

Heller, K. A. and Ghahramani, Z. (2005) Bayesian hierarchical clustering. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 297–304. New York, NY, USA: ACM.

Jasra, A., Stephens, D. A. and Holmes, C. C. (2007) Population-Based Reversible Jump Markov Chain Monte Carlo. *Biometrika*, **94**, 787–807.

Johnson, S. C. (1967) Hierarchical clustering schemes. *Psychometrika*, **32**, 241–254.

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967*, **1**, 281–297.

Ramoni, M., Sebastiani, P. and Kohane, P. R. (2002) Cluster analysis of gene expression dynamics. *Proc. Nat. Acad. Sci. USA*, **99**, 9121–9126.

Richardson, S. and Green, P. J. (1997) On Bayesian analysis of mixtures with an unknown number of components (with discussion). *J. Roy. Statist. Soc. B*, **59**, 731–792.

Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D. and Futcher, B. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, **9**, 3273–3297.

Teh, Y. W., Daumé, III, H. and Roy, D. (2008) Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems*, **20**, 1473–1480.

Ward, J. H. (1963) Hierarchical grouping to optimize an objective function. *J. Amer. Statist. Assoc.*, **58**, 236–244.

Yeung, K. Y., Fraley, C., Murua, A., Raftery, A. E. and Ruzzo, W. L. (2001) Model-based clustering and data transformations for gene expression data. *Bioinformatics*, **17**, 977–987.

Data: A collection of $n \geq 2$ data points $y_1, \ldots, y_n$.
Result: A partition $\mathfrak{C}$ of the n data points and a binary tree $\mathfrak{T}$ which passes through $\mathfrak{C}$.

Set $k = n$. Let the current clustering configuration be made up of the singleton sets, $\mathfrak{C} = \{\{y_1\}, \ldots, \{y_n\}\}$. Let π_n be the objective function value for $\mathfrak{C}$, $\pi_n = \pi(\mathfrak{C})$.
for $1 \leq i < n$ **do**
  for $i < j \leq n$ **do**
    Let $\mathfrak{C}^{(ij)}$ be the clustering that would result from merging clusters i and j in $\mathfrak{C}$.
    Calculate the ij^{th} entry s_{ij} of a symmetric $n \times n$ similarity matrix S given by

$$s_{ij} = s_{ji} = \frac{\pi(\mathfrak{C}^{(ij)})}{\pi(\mathfrak{C})} \tag{1}$$

  end
end
while $k > 1$ **do**
  for $1 \leq i \leq k$ **do**
    Identify the closest other cluster i^* in $\mathfrak{C}$ to cluster i according to the measure (1),

$$i^* = \underset{j}{\operatorname{argmax}}\ s_{ij}. \tag{2}$$

    Let $s_i = s_{ii^*}$ be the corresponding maximized similarity value for cluster i.
  end
  Set $\hat{i} = \operatorname{argmax}_i s_i$.
  Merge cluster $\hat{i}$ with cluster $\hat{i}^*$ to form a new cluster $\hat{i}$. Let $\mathfrak{C}$ now refer to this new clustering, relabeling the other clusters accordingly.
  Set $k = k - 1$. Let $\Delta_M = \emptyset$.
  Set $\Delta = \Delta_M \cup \{\hat{i}\}$.
  Calculate $\pi_k = \pi(\mathfrak{C})$.
  foreach $i \in \Delta$ **do**
    foreach $j \notin \Delta$ **do**
      Calculate $s_{ij} = s_{ji}$, the similarity of cluster i to cluster j, and hence identify the new nearest cluster to i, i^* and possibly to j, j^*.
    end
  end
end
Looking back over the partitions visited, find the number of clusters k in the hierarchy maximizing the objective function, $k = \operatorname{argmax}_k\ \pi_k$. Let $\mathfrak{C}$ be that partition.
Report $\mathfrak{C}$ as the output clustering.

Algorithm 1: Agglomerative clustering (AC)

Data: A collection of $n \geq 2$ data points $y_1, \ldots, y_n$; a partition $\mathfrak{C}$ of the n data points.
Result: A new partition $\mathfrak{C}'$ of the data points.

Let Δ_M be the set of clusters which are affected by this algorithm, initialized so $\Delta_M = \emptyset$.
Let k be the number of clusters in $\mathfrak{C}$. Arbitrarily label the clusters $\{1, \ldots, k\}$.
Let M be a set of subjects that are suboptimally located, initialized so $M = \emptyset$.
for $1 \leq i < n$ **do**
    Let $c(i)$ be the index of the cluster in $\mathfrak{C}$ containing subject i.
    for $1 < j \leq k + 1$ **do**
        Let $\mathfrak{C}^{(i \to j)}$ be the clustering that would result from moving subject i into cluster j in $\mathfrak{C}$ (where $\mathfrak{C}^{(i \to k+1)}$ is the clustering that would result from moving subject i into its own singleton cluster.)
        Calculate the ij^{th} entry r_{ij} of a $n \times k$ subject-cluster similarity matrix R given by
        $r_{ij} = \pi(\mathfrak{C}^{(i \to j)})$
    end
    Set $c'(i) = \text{argmax}_j \, r_{ij}$, and $r_i = r_{ic'(i)}$.
    If $c'(i) \neq c(i)$, set $M = M \cup \{i\}$.
end
Let $i^* = \text{argmax}_{i \in M} \, r_i$. Set $\Delta_M = \Delta_M \cup \{c(i^*), c'(i^*)\}$. Set $c(i^*) = c'(i^*)$, so i^* moves to its optimal cluster $c'(i^*)$.
Repeat steps 2-14 until $M = \emptyset$ so that all subjects are in their optimal cluster. Let $\mathfrak{C}'$ be this final clustering.

Algorithm 2: Sequential reclassification algorithm

Data: A collection of $n \geq 2$ data points $y_1, \ldots, y_n$; a partition $\mathfrak{C}^*$ of the n data points.
Result: A new partition $\mathfrak{C}$ of the data points and a binary tree $\mathfrak{T}$ which passes through $\mathfrak{C}$.

Perform Algorithm 1 on the data $y_1, \ldots, y_n$ with the following alteration to the similarity measure (1): Let $s_{ij} = -\infty$ for any clusters i and j when each are subsets of different clusters in $\mathfrak{C}^*$, with at least one a proper subset.

Algorithm 3: Constrained agglomerative clustering algorithm

Identical to Algorithm 1, but with the following alteration:
Randomly set $\hat{i} = i$ with probability $s_i / \sum_{j=1}^{k} s_j$.

Algorithm 4: Stochastic agglomerative clustering algorithm (SAC)

Algorithm	$\log(\pi(\mathfrak{C}))$	$\log(\bar{\pi}_{\mathfrak{T}}(\mathfrak{C}))$	$\log(\pi(\mathfrak{T}))$	Purity	Time (s)	# Models
AC	-420.149	-424.909	-669.594	0.761	0.1	39,602
BHC	-387.225	-392.081	-628.906	0.878	0.1	39,602
ACIR	-339.423	-344.138	-627.846	0.963	0.6	67,548
AIRC	-337.476	-342.393	-629.952	0.950	0.2	132,673
SACIR	-337.839	-342.649	-628.424	0.960	0.2	49,512
SAIRC	-339.010	-343.673	-627.373	0.972	0.2	110,924
BHCIR	-338.757	-343.684	-638.151	0.920	0.5	66,390
ACIR2	-339.367	-344.140	-628.492	0.962	2.7	814,251
AIRC2	-337.105	-342.023	-628.906	0.960	1.6	896,229
SACIR2	-337.839	-342.649	-628.424	0.960	1.4	709,275
SAIRC2	-337.105	-342.023	-628.906	0.960	1.6	827,004
BHCIR2	-338.757	-343.684	-638.151	0.920	1.8	700,661

Table 1: Quantitative performance of clustering algorithms on the synthetic normal data. For each algorithm is shown the log of the highest model probability found ($\pi(\mathfrak{C})$), the log average model probability across the agglomerative tree ($\bar{\pi}_{\mathfrak{T}}(\mathfrak{C})$), the log BHC score of Heller and Ghahramani (2005) for the entire tree ($\pi(\mathfrak{T})$), the purity, the run time to compute and the number of partitions evaluated by the algorithm.

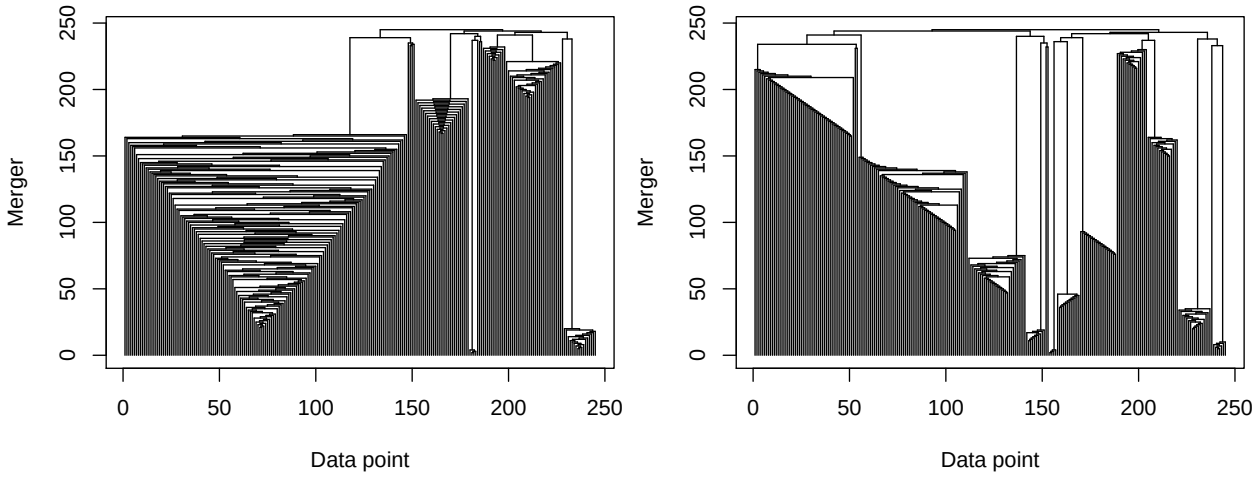

Figure 2: Dendrogram output for the Enzyme data from the clustering algorithms AC (left) and AIRC2 (right).

Data set	Algorithm	$\log(\pi(\mathfrak{C}))$	$\log(\bar{\pi_{\mathfrak{T}}}(\mathfrak{C}))$	k	Time (s)	# Models
Galaxy ($n = 82$)	AC	-92.518	-96.506	9	0.03	6,562
	BHC	-97.481	-101.564	9	0.02	6,562
	ACIR	-90.053	-94.113	9	0.04	9,309
	AIRC	-90.053	-94.113	9	0.05	16,428
	SACIR	-90.672	-94.507	8	0.04	7,882
	SAIRC	-90.053	-94.113	9	0.05	20,367
	BHCIR	-90.719	-94.825	9	0.04	8,583
	ACIR2	-89.864	-94.000	9	0.16	143,480
	AIRC2	-89.864	-94.000	9	0.14	155,411
	SACIR2	-90.672	-94.507	8	0.18	156,584
	SAIRC2	-89.864	-94.000	9	0.14	158,223
Acidity ($n = 155$)	AC	57.851	52.972	7	0.04	23,717
	BHC	68.867	63.896	12	0.04	23,717
	ACIR	76.234	71.340	7	0.07	32,283
	AIRC	81.386	76.693	9	0.11	80,673
	SACIR	87.541	83.631	12	0.08	34,660
	SAIRC	87.699	83.611	12	0.12	79,234
	BHCIR	88.758	83.937	10	0.08	35,259
	ACIR2	80.651	75.743	8	0.64	584,655
	AIRC2	89.484	84.585	9	0.66	953,191
	SACIR2	87.886	83.974	12	0.61	706,201
	SAIRC2	87.710	83.557	11	0.48	722,309
Enzyme ($n = 245$)	AC	266.975	261.887	6	0.07	59,537
	BHC	297.910	292.884	20	0.08	59,537
	ACIR	380.345	375.188	10	0.38	192,323
	AIRC	390.145	384.790	11	0.32	259,863
	SACIR	386.863	381.788	11	0.25	128,631
	SAIRC	388.034	382.755	11	0.25	187,857
	BHCIR	389.140	383.777	11	0.19	115,568
	ACIR2	380.345	375.188	10	1.68	1,608,924
	AIRC2	390.202	384.841	11	1.68	2,349,733
	SACIR2	387.769	382.694	11	3.12	3,435,005
	SAIRC2	389.152	383.790	11	1.60	2,121,777

Table 2: Quantitative performance of clustering algorithms on the Galaxy, Acidity and Enzyme data. For each algorithm is shown the log of both the highest model probability found ($\pi(\mathfrak{C})$) and the average model probability across the agglomerative tree ($\bar{\pi_{\mathfrak{T}}}(\mathfrak{C})$), the estimated number of clusters k, the run time to compute and the number of partitions evaluated by the algorithm.

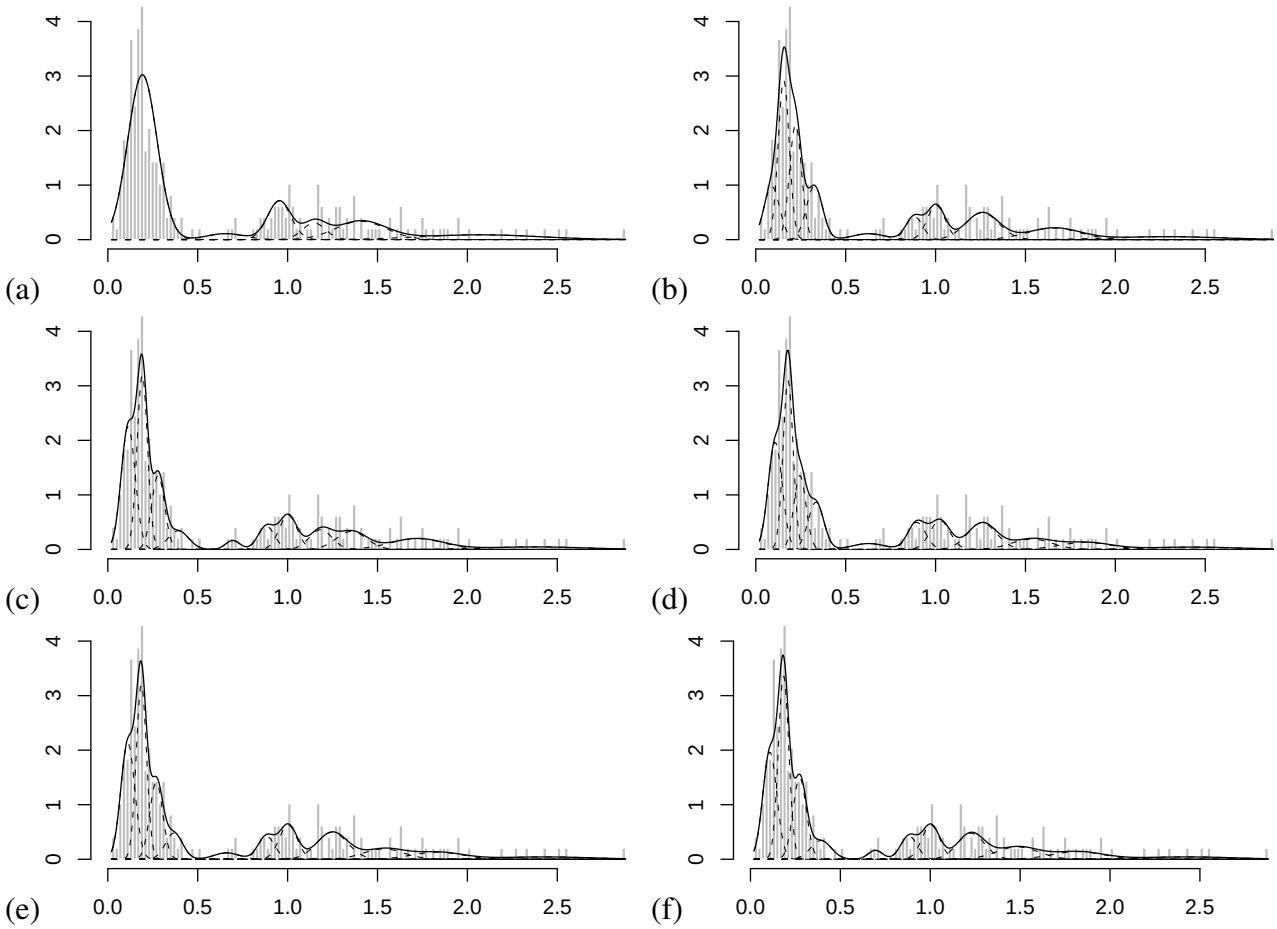

Figure 3: Histogram for the Enzyme data and predictive densities of optimal models found from algorithms (a) AC; (b) ACIR; (c) AIRC; (d) SACIR; (e) SAIRC; (f) BHCIR. The dashed lines represent the weighted conditional predictive densities for each of the clusters found, and the solid line the overall marginal predictive density.

Algorithm	$\log(\pi(\mathfrak{C}))$	$\log(\bar{\pi}_{\mathfrak{T}}(\mathfrak{C}))$	k	Time (s)	Reallocations/Subjects	# Models
AC	-3192.26	-3199.97	24	5	-	7,672,901
BHC	-5567.94	-5575.46	7	10	-	7,672,901
ACIR	-2757.48	-2764.91	24	23	1,036/628	13,818,776
AIRC	-2702.50	-2710.28	24	179	7,110/2,216	100,996,783
SACIR	-2750.76	-2758.34	20	23	1,052/600	14,050,474
SAIRC	-2686.78	-2694.56	25	231	8,498/2,386	65,240,667
BHCIR	-2814.88	-2822.80	18	66	3,142/1,724	27,566,487
ACIR2	-2753.03	-2760.82	25	709	7,259/1,237	412,057,319
AIRC2	-2702.50	-2710.28	24	613	10,387/2,383	346,651,897
SACIR2	-2737.99	-2745.73	22	619	8,765/1,299	336,607,411
SAIRC2	-2686.71	-2694.50	25	722	11,977/2,527	418,788,376

Table 3: MAP model scores and number of clusters k found for the gene expression data (2,771 points). Run times given in seconds. The penultimate column shows the number of reallocations performed by each algorithm and the number of subjects which were involved in these reallocations. The final column gives the total number of partitions evaluated throughout the course of the algorithm.

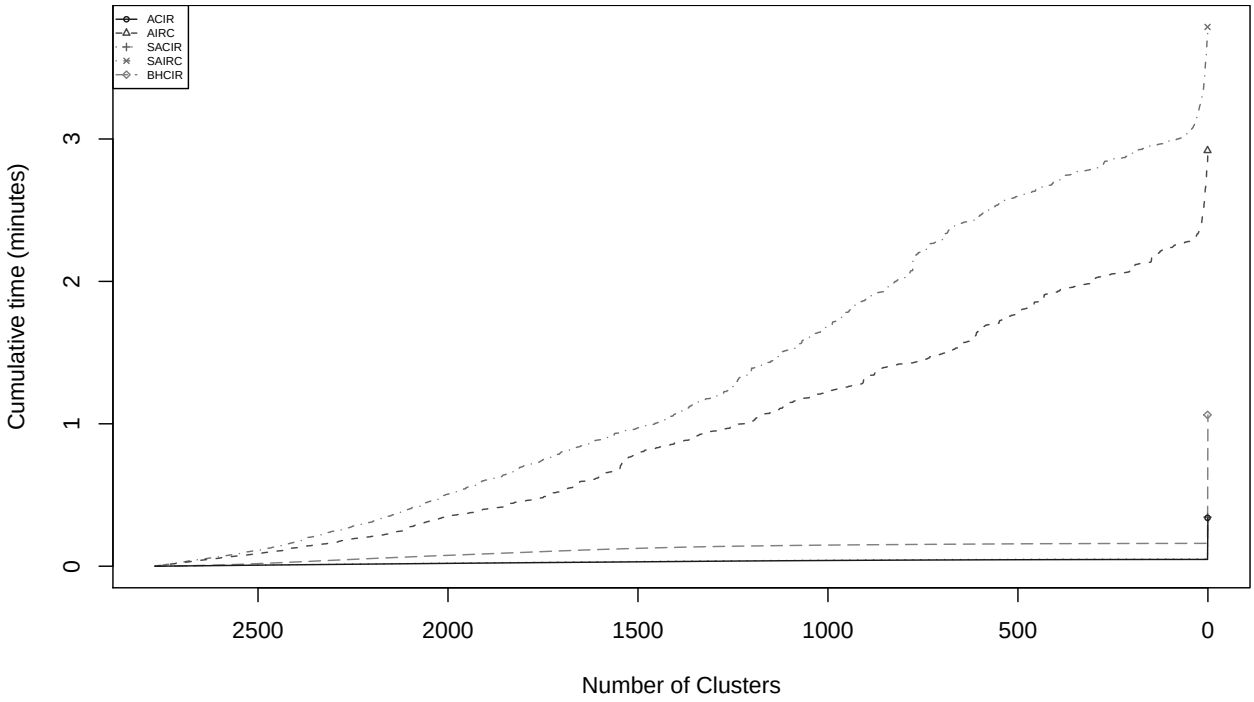

Figure 4: The cumulative time taken to agglomerate the gene expression data to each level under the different clustering algorithms.

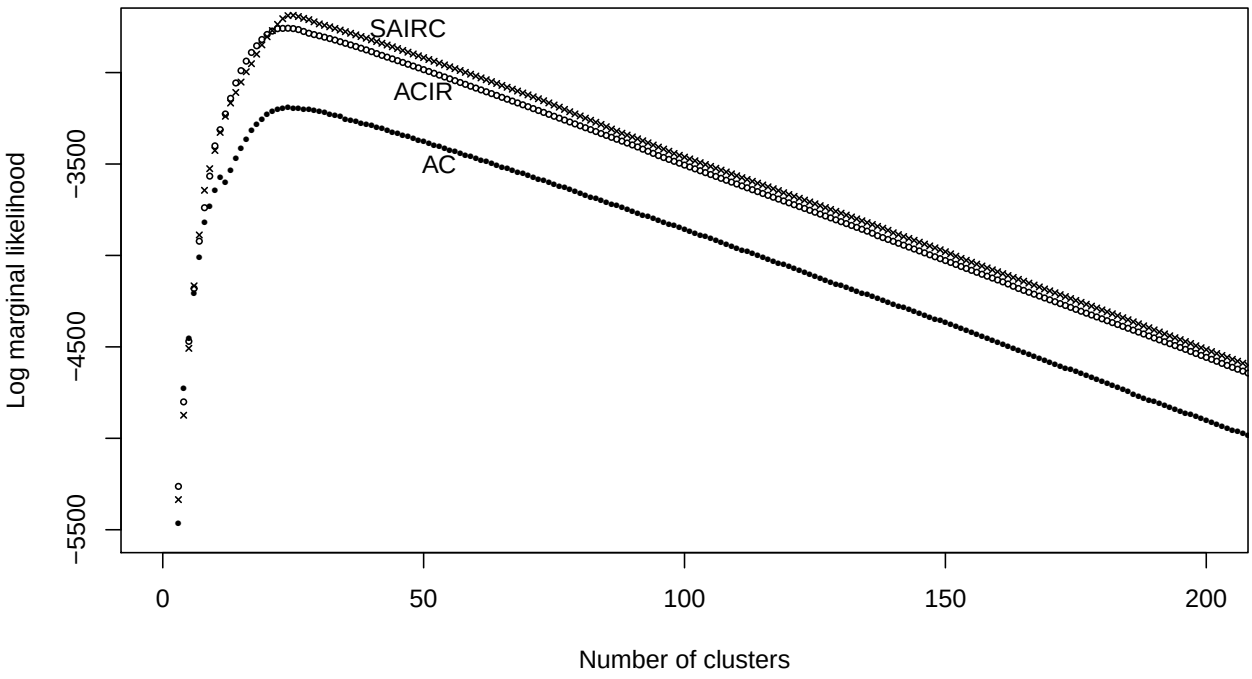

Figure 5: Objective function profile under hierarchical clustering through AC, ACIR and SAIRC.