Search techniques for gaussian mixture modelling with Minimum Message Length, with applications to the origin of our Galaxy

Aldeida Aleti[b,*], Andrew R. Casey[a,b], John C. Lattanzio[a], David L. Dowe[b]

[a] *School of Physics and Astronomy, Monash University, Melbourne, Clayton VIC 3800, Australia*
[b] *Faculty of Information Technology, Monash University, Melbourne, Clayton VIC 3800, Australia*

Abstract

Keywords: Search, Minimum Message Length, MML, gaussian mixture models, Galactic Archaeology

1. Introduction

The contributions of this paper are:

- Solving the galactic archaeology (GA) problem using mixture modelling
- Minimum Message Length (MML) formulation
- $\mathcal{O}(K)$ search methods for gaussian mixture modelling

2. Galactic Archaeology

Galactic Archaeology [1] aims at reconstructing the history of our galaxy by identifying ancient star formation events. Most stars are born in clusters (of order 10^0 to 10^6 stars at a time), and all approximately share the same chemical composition of the gas cloud that they form from. Stars in clusters often do not remain gravitationally bound, and disperse throughout the disk of our galaxy within about 10^9 yr. As a consequence, where a star is now does not reflect its formation site, which complicates our understanding of galaxy formation and evolution. However, the observable chemical composition of a star remains largely the same throughout that star's lifetime, providing an immutable marker (a chemical tag, or 'fingerprint') that carries with it information about where and when every star formed.

*Corresponding author
Email address: aldeida.aleti@monash.edu (Aldeida Aleti)

Observations reveal that stars in surviving clusters show very little scatter in their chemical composition [2], and that star clusters can be identified solely on the basis of stellar chemical abundances [3]. Astronomers have precisely measured the chemical composition for hundreds of thousands of stars in the disk of our galaxy, where the natal formation site for most stars is not known [4]. Indeed, from those data it is not known how many star formation sites we might expect to be able to detect, and we do not know the number of stars that we might to observe from each cluster (the so-called stellar mass function). Theoretical simulations can provide a guide to these quantities, but the predictions are still uncertain.

These facts conspire to produce a challenging parameter estimation and model selection problem that is analagous to mixture modelling, where the true number of mixtures is unknown. Little work has been done to address this problem so far, apart from the work of [5], who used the *Manhattan distance* metric to quantify the chemical difference between any two stars, and estimate the level of homogeneity expected within a cluster. This metric calculates the mean weighted absolute difference of abundance levels, σ_C, between two stars S^a and S^b as follows: $\sigma_C = \sum_{c \in C} w_c |S_c^a - S_c^b| / ||C||$, where C is the set of chemical species, S_c^a is the chemical abundance of element c in star S^a, w_c is the weight given to species c, and $||C||$ is the total number of chemical species[1]. The Manhattan distance values are high for stars from different clusters, and low for stars from the same cluster. Note that this metric is not particularly robust or versatile; for example it is not invariant even to a simple rotation of the co-ordinate axes. An important issue is how to set the weights for each chemical species w_c, and how they affect the results (Mitschang et al. [5] used $w_c = 1$). Another issue is setting the boundary between clusters. It is not clear what a high and low Manhattan distance is; these values are subjective. In this paper, we express the problem in terms of mixture modelling and infer the number of clusters, their mixing proportions (or relative abundances) and each cluster's statistical distribution using Minimum Message Length (MML) and novel search techniques that are readily applicable to the large data sets being accumulated by astronomers worldwide.

3. Related Work in Mixture Modelling

Mixture modelling is composed of three main parts:

1. an estimator of component parameters of a mixture;
2. an objective functions that scores a mixture; and
3. a search strategy which identifies the best number of components and their relative weights.

[1] Here we describe the work of Mitschang et al. (2014) [5] using their notation, but we caution that we use similar notation in this work with very different terminology and meaning.

Different criteria for estimating component parameters have been used in the literature, with traditional methods using maximum likelihood estimation (MLE) [6] or maximum a posteriori probability (MAP) [7]. These methods do not consider the complexity of the model, but only focus on the goodness-of-fit, and require modification to minimise the risk of over-fitting. In this work, we focus on the Bayesian information-theoretic Minimum Message Length (MML) principle [8, 9, 10, 11]. Unlike MLE or MAP estimators, the MML principle is invariant under non-linar data transformations, and can seemlessly incorporate different probability distributions.

The classically-described princple of MML is that the best explanation of the data is the one that leads to the shortest so-called two-part message [11], where a *message* takes into account both the complexity of the model and its explanatory power. The complexity of the model is described through the first part of the message, and the second part of the message describes its explanatory power. The *length* of each message part is quantified (or measured) using information theory, allowing for a fair evaluation between different models of varying complexity or explanatory power. MML has been shown to perform well on a variety of empirical analyses (see, e.g., [12, 13], with references to further examples in [11, 14, 15, 16]).

Arguments about the statistical consistency (i.e., as the number of data points increases the distributions of the estimates become increasingly concentrated near the true value) of MML are given in [17, 16]. The MML principle requires that we explicitly specify our prior beliefs on the model parameters, providing a Bayesian optimisation approach which can be applied across entire classes of models.

Two other well-known methods based on information theory are the Akaike Information Criterion (AIC) [18] and Bayesian Information Criterion (BIC) [19], both of which endeavour to temper maximum likelihood's tendency to over-fit and under-estimate the noise. Unlike maximum likelihood, MML accounts for model complexity, and for this reason maximum likelihood can be thought of as MML without including the first part of the message that describes the model complexity. AIC [18] chooses a model with the motivation of minimizing the Kullback-Leibler divergence between the model and the truth:

$$AIC = -2 \log_e f(\mathbf{y}|\boldsymbol{\theta}) + 2k \tag{1}$$

where the likelihood $f(\mathbf{y}|\boldsymbol{\theta})$ is the probability of the data $\mathbf{y}$ given a model parameterised by $\boldsymbol{\theta}$, and k is the number of free parameters in the model. BIC [19] uses a constant penalty equal to $\frac{1}{2} \log N$ for each free parameter in the model, where N is the number of data points. Minimum Description Length (MDL) [20] is a similar approach to BIC [19], and has close (albeit also distinct) ties to MML [8]. AIC and BIC/MDL associate parameter costs (or model complexity) with the number of free parameters, not the values of those parameters or the variance that can be explained by those parameters. As a result, when using AIC or BIC/MDL formalisms, all models of a particular probability distribution share the same cost, regardless of their respective means, covariance matrices, or the Fisher information, which could bias inferences.

The parameters of a mixture model can be estimated using Expectation-Maximisation (EM) [21]. The EM algorithm has been used with MML to infer Gaussian mixtures under different simplifying assumptions, such as assuming that the covariance matrices are diagonal [22, 23, 24] (where it is sometimes called an 'adjust cycle') and [25], or approximating the probabilities of mixture parameters [26, 27]. The search algorithm employed in the majority of these approaches is based on running EM iteratively for different numbers of components [22, 23, 24, 25, 27, 28]. The search method proposed by Figueiredo and Jain [26], on the other hand, starts with a large number of components, and iteratively deletes components. A mirror extension of this method was proposed by Kasarapu and Allison [29], which starts with one components, and selectively splits, deletes, or merges components in an attempt to minimise the message length. While this method removes the overhead introduced by the search algorithm of Figueiredo and Jain [26], the search remains greedy, and can be prone to premature convergence in suboptimal solutions. The search methods introduced in this paper, incorporate a perturbation step which encourages explorations of new local optima.

4. Minimum Message Length (MML) for mixture modelling

A complex model is not an optimal one, unless its complexity is justifiable by the added explanatory power. There is an established connection between simplicity and truth, as beautifully elucidated by the Occam's Razor principle. MML is a statistical technique that attempts to formalize and quantify this principle.

The *message* must encode two parts: the model, and the data given the model. The encoding of the message is based on Shannon's information theory [30]. The information gained from an event e occurring, where $p(e)$ is the probability of that event, is $I(e) = -\log_2 p(e)$. The information content is largest for improbable outcomes, and smallest for outcomes that we are almost certain about. In other words, an outcome that has a probability close to unity has zero information content because nothing new is learned from it, whereas rarer events convey a much higher information content.

Formally, the length of the first part of the message is $-\log_2 p(\boldsymbol{\theta})$, and that of the second part is $-\log_2 f(\mathbf{y}|\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ is the prior probability of model described with parameters $\boldsymbol{\theta}$, and $f(\mathbf{y}|\boldsymbol{\theta})$ is the likelihood of data $\mathbf{y}$ given a model parameterised by $\boldsymbol{\theta}$. The total message length is the sum of the two parts, i.e. $-\log_2 p(\boldsymbol{\theta}) - \log_2 f(\mathbf{y}|\boldsymbol{\theta})$, which is minimised precisely when $p(\boldsymbol{\theta})\, f(\mathbf{y}|\boldsymbol{\theta})$ is maximised. The multiplication of the prior and the likelihood is proportional to the posterior *probability* $g(\boldsymbol{\theta}|\mathbf{y})$. An important but subtle point is that, within the MML framework, we are quantising (or rounding off in) parameter space, and so $p(\boldsymbol{\theta})$ corresponds to a *probability* and *not* a density. Thus, the MML estimate can in some sense be thought of as maximising $p(\boldsymbol{\theta})\, f(\mathbf{y}|\boldsymbol{\theta})$ and in turn (in some sense) equal to the posterior mode. This is similar in spirit to the (Bayesian) Maximum A Posteriori (MAP) posterior mode, partly because both

terms $-\log_2 p(\boldsymbol{\theta})$ and $\log_2 f(\mathbf{y}|\boldsymbol{\theta})$ here correspond to probabilities, but with the benefit of statistical invariance under re-parameterisation (e.g., [9, 10, 11, 14]).

4.1. Component message lengths

Mixture modelling allows for the partitioning of data into overlapping groups. The MML principle requires that we encode everything required to reconstruct the message, including our prior beliefs on the model parameters. Consider that we have N data points each with D dimensions which are to be modelled by a finite number of K Gaussian mixtures. Within the context of using the MML principle for mixture modelling, the *message* would have to specifically encode:

(1) The model

a. The number of mixture components (or groups) K.

b. The relative weight w_k (or contribution, or mixing proportion) of the amount(s) of data in each group.[2]

c. The parameters of each component, specifically the mean $\boldsymbol{\mu}_k$ and covariance matrix $\mathbf{C}_k$ for all K Gaussian components.

(2) The data given the model – the likelihood – where the probability density function f for a multivariate normal distribution with D dimensions is given by

$$f(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^D |\mathbf{C}|}} \exp\left[-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\intercal \mathbf{C}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right] \tag{2}$$

and for a mixture of K gaussian mixtures this becomes

$$f(\mathbf{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} w_k f(\mathbf{y}|\boldsymbol{\theta}_k) \quad . \tag{3}$$

Such that the log-likelihood $\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$ is

$$\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} w_k f(\mathbf{y}_n|\boldsymbol{\mu}_k, \mathbf{C}_k) \quad . \tag{4}$$

Calculating the length of the message can be a non-trivial task, especially if the model is reasonably complex. This makes the MML principle intractable (or uncomputable) in most cases, and forces us to make approximations when calculating the message length. Using a Taylor expansion, a generalised scheme can be derived to estimate the parameter vector $\boldsymbol{\theta}$ that minimises the message length expression[9]:

[2] In practice only $K-1$ weights must be encoded because $\sum w_k = 1$.

$$I(\boldsymbol{\theta}, \mathbf{y}) = \frac{Q}{2} \log \kappa(Q) - \log \left(\frac{p\left(\boldsymbol{\theta}\right)}{\sqrt{|\mathcal{F}\left(\boldsymbol{\theta}\right)|}} \right) - \log \mathcal{L}\left(\mathbf{y}|\boldsymbol{\theta}\right) + \frac{Q}{2} \quad . \tag{5}$$

Here $p\left(\boldsymbol{\theta}\right)$ is the prior, Q is the number of free parameters in the model,

$$Q = \frac{KD(D+3)}{2} + K - 1 \quad , \tag{6}$$

and $\kappa(Q)$ is a function to approximate the lattice quantisation constant for Q free parameters[31], and $|\mathcal{F}\left(\boldsymbol{\theta}\right)|$ is the determinant of the *expected* Fisher information matrix (the second order partial derivatives of the negative log-likelihood function $-\log \mathcal{L}\left(\mathbf{y}|\boldsymbol{\theta}\right)$). A common issue that arises after listing the components above is that although we have explicitly encoded the model parameters, we have not encoded 'the model' in the sense that we have not described how these encoded parameters relate to each other. In a sense, we do not describe to the receiver that these encoded parameters refer to a mixture of Gaussians, or what we mean when we refer to the term 'Gaussian'. This information is implicit, and justified if we are evaluating and comparing models within the same class (e.g., gaussian mixture models). For example, it is analogous to writing the probability of some data given a model as $p(\mathbf{y}|\boldsymbol{\theta})$ rather than what is actually implied: $p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\mathcal{M}})$, the probability of the data given the model parameters $\boldsymbol{\theta}$ and class of model $\boldsymbol{\mathcal{M}}$. Whatever the message length required to encode to class of model $\boldsymbol{\mathcal{M}}$, under consideration, it would be a constant added to the message length regardless of the number of components in our mixture.

In Appendix Appendix A we define our priors and describe the individual contributions to the mixture message length. For the sake of brevity we denote the full set of model parameters as $\boldsymbol{\psi} = \{K, \mathbf{w}, \boldsymbol{\mu}, \mathbf{C}\}$. The total message length $I(\mathbf{y}|\boldsymbol{\psi}, N, D)$ to model some $N \times D$ dimensional data $\mathbf{y}$ with a mixture of multivariate Gaussian components is given by:

$$\begin{aligned} I(\mathbf{y}|\boldsymbol{\psi}, N, D) \quad = \quad & -\log \mathcal{L}\left(\mathbf{y}|\boldsymbol{\theta}\right) - \frac{(D+2)}{2} \sum_{k=1}^{K} \log |C_k| \\ & \cdots + \left(\frac{D(D+3)}{4} - \frac{1}{2} \right) \sum_{k=1}^{K} \log w_k + K \left(1 - \frac{D}{2} \right) \log 2 \\ & \cdots + \log \Gamma(K) + \frac{1}{2} \left(\log Q\pi - Q \log 2\pi \right) + \frac{Q}{2} \log N \quad . \end{aligned} \tag{7}$$

4.2. Expectation-Maximization

Given a mixture model with K components, we iteratively update the model parameters $\boldsymbol{\theta}$ by expectation-maximization [21]. In the expectation step we calculate the partial memberships of the data to each component, as specified by the responsibility matrix

$$\mathcal{R}_{nk} = \frac{w_k f(\mathbf{y}_n|\boldsymbol{\mu}_k, \mathbf{C}_k)}{\sum_{k=1}^{K} w_k f(\mathbf{y}_n|\boldsymbol{\mu}_k, \mathbf{C}_k)} \tag{8}$$

where $\mathcal{R}_{nk}$ represents the fractional assignment of the nth data point to the kth component. At the maximization step we update our estimates of the mixture parameters to minimize the total message length (Equation A.16). The updated estimates for the relative weights are obtained by differentiating Equation A.16 with respect to w_k under the constraint $\sum w_k = 1$ (see [29] for derivation) such that:

$$w_k^{(t+1)} = \frac{2n_k^{(t)} + 1}{2N + K} \tag{9}$$

where n_k is the effective membership for the kth component

$$n_k = \sum_{i=1}^{N} \mathcal{R}_{nk} \quad . \tag{10}$$

The updates of the mean and covariance matrix are given by:

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{N} \mathcal{R}_{nk}^{(t)} \boldsymbol{\mu}_k \tag{11}$$

$$\mathbf{C}_k^{(t+1)} = \frac{1}{n_k^{(t)} - 1} \sum_{i=1}^{N} \mathcal{R}_{nk}^{(t)} \left(\mathbf{y}_i - \boldsymbol{\mu}_k^{(t+1)}\right) \left(\mathbf{y}_i - \boldsymbol{\mu}_k^{(t+1)}\right)^{\intercal} \quad . \tag{12}$$

Throughout this work we consider the mixture to be converged when the change in message length is less than 10^{-5} nats.

4.3. Predicting the message length of future mixtures

The objective function in Equation A.16 is valid for a class of Gaussian mixture models, and the EM updates that act to minimize Equation A.16 are valid for a mixture with a given number of K components. However, we require a search strategy to minimise Equation A.16 across all possible numbers of components. Existing search strategies will greedily seek to trial different numbers of components. Here we describe methods to predict the minimum message length of a future (untrialled) mixture, which help to guide our search strategies (Section 5). We separately describe the predictions for different items in Equation A.16:

1. predicting the sum of the log of the weights;
2. predicting the negative log-likelihood; and
3. predicting the sum of the log of the determinant of the covariance matrices.

These items directly contribute to Equation A.16, and their values are not known before trialling a mixture. All other contributions to Equation A.16 can be calculated directly given K, N, and D.

4.3.1. Predicting $\sum \log w_k$

We consider the two extremes of $\sum \log w_k$ for a mixture of K components:

1. The weights are distributed uniformly such that $w_k = 1/K$ and

$$\sum_{k=1}^{K} \log w_k = \sum_{k=1}^{K} \log \frac{1}{K} = -K \log K \quad . \tag{13}$$

2. The weights are distributed such that $K-1$ components each describe just two data points $w_{2..K} = 2/N$, and the remaining component describes all other data points $w_1 = 1 - 2(K-1)/N$ as $\sum_{k=1}^{K} w_k = 1$. At this extreme,

$$\sum_{k=1}^{K} \log w_k = 2(K-1) \log \frac{2}{N} + \log \left(1 - \frac{2(K-1)}{N}\right) \tag{14}$$

and by making use of logarithmic identities:

$$\sum_{k=1}^{K} \log w_k = (K-1) \log 2 - K \log N + \log (N - 2K + 2) \tag{15}$$

Thus, for any finite mixture of K Gaussian components we have the bounds

$$-K \log K \quad \geq \quad \sum_{k=1}^{K} \log w_k \quad \geq \quad (K-1) \log 2 - K \log N + \log (N - 2K + 2) \tag{16}$$

which describe the smallest and largest contribution that the weights can contribute to the message length for any untrialled mixture with K components. When making predictions for future mixtures, we record the optimal weights for each trialled mixture and calculate a *uniformity fraction* $f_u \in [0, 1]$

$$f_u = \frac{\alpha - \sum \log w_k}{\alpha + K \log K} \tag{17}$$

where for brevity $\alpha = (K-1) \log 2 + \log(N - 2K + 2) - K \log N$, and f_u the uniformity of the mixture from Equation **??** ($f_u = 0$) to Equation 14 ($f_u = 1$). We use estimates of f_u from previously trialled mixtures to make predictions for the uniformity of future, untrialled mixtures, and the message length required to encode the weights. Given the theoretical bounds in Equations 14 and **??**, we clip the predictions of the expectation value on the message length required to encode the weights by these bounds.

4.3.2. Predicting $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$

The negative log-likelihood could be infinitely high given the wrong component parameters. When predicting the negative log likelihood $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$ we seek to predict the *minimum* $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$ for an untrialled mixture with K

components. We assume that the improvement in $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta}$ between subsequent trials of K improves of order $1/K$ [**?**]. We assume a mean model for the minimal message lengths I of previously trialled mixtures with K components as

$$-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta}) = \frac{\alpha}{K} + \beta \tag{18}$$

where α and β are unknowns, and fit these data using a Gaussian Process [**?**] where the covariance between the data (K, I) is modelled by a squared exponential kernel function [**?**] with white noise component added to the diagonal of the covariance matrix. We simultaneously optimized the parameters α, β and the kernel hyperparameters [**?** **?**]. Given this Gaussian Process model with the optimized parameters, we make predictions for the minimum message length of untrialled mixtures. Specifically we are making predictions for the expectation value of the minimum message length of untrialled mixtures, and each prediction of the expectation value has an associated variance in its prediction. Given the kernel adopted, the predicted variance will be higher for untrialled mixtures that are farthest away (in K) than any trialled mixture (Figure **??**).

4.3.3. Predicting $\sum \log |\mathbf{C}_k|$

The remaining component whose contribution to the message length is unknown is the sum of the log of the determinant of the covariance matrices $\sum \log |\mathbf{C}_k|$. For most practical scenarios this component is a much smaller order of magnitude than the negative log likelihood $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$. Through experiments with generated data sets, we found that the message length to encode $\sum \log |\mathbf{C}_k|$

$$I_{sldc} = -\frac{1}{2}(D+2)\sum \log |\mathbf{C}_k| \tag{19}$$

tended to decrease with increasing K components when the value of K is low (e.g., $\approx \frac{1}{5}K_{true}$), and then began to increase linearly with K until K_{true}. Our experiments suggested that this is merely a consequence of the search process, and cannot be reliably used for a heuristic for predicting K_{true}, as the turnover point occurred when the first true data component is well-described by a single model component. That is to say that by $K \approx \frac{1}{5}K_{true}$, there is one component covariance matrix that will no longer change significantly, and therefore the determinants of the component covariance matrices that are added to the mixture will only tend to decrease on average, producing a near linear increase in I_{sldc}. Before the turnover point, as components are added the average determinants of the covariance matrices continues to increase.

We trialled several heuristics to approximate the change in $\sum \log |\mathbf{C}_k|$ between mixtures. While some heuristics based on the distribution of distances between data points appeared promising, we found good results by simply modelling I_{sldc} as a function of the number of trialled components K using a Gaussian Process. We used a Matérn-3/2 kernel [**?** **?**] function to model the covariance between data points (K, I_{sldc}) and added hyperparameters to fit the data mean, and a white noise component that was added to the diagonal of the

covariance matrix. We optimized these hyperparameters using the L-BFGS-B algorithm [**?** **?** **?**], and made predictions for the message length required to encode the sum of the log of the determinant of covariance matrices of untrialled mixtures.

Like the predictions from our Gaussian Process model for $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$, our predictions for the expectation value of $-\frac{1}{2}(D+2)\sum \log |\mathbf{C}_k|$ (for the optimal mixture) come with an associated variance on that prediction.

4.3.4. Combining the predictions

We sum the predictions in the message length required to describe $\sum \log w_k$, the negative log likelihood $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$, and the sum of the log of the determinant of the covariance matrices $\sum \log |\mathbf{C}_k|$, as well as constant and deterministic terms, to predict the (minimum) message length of an untrialled mixture. We sum the variances in the predictions of each component. This process is illustrated in Figure **??** where we show the predicted (blue) and actual message lengths (black) for various mixtures. The minimum message length over all models $\boldsymbol{\mathcal{M}}$ is predicted to be near $K \approx X$, even though only a few mixtures have been trialled, and the nearest trialled mixture is $K = X$.

5. Search Strategies

We introduce and evaluate two search strategies that make use of the objective function in Equation A.16. These strategies are benchmarked against an existing state-of-the-art search method, which we have adapted to use the same objective function in Equation A.16 in order to facilitate a fair comparison.

5.1. Benchmark Method

We use the search algorithm proposed [29] as a benchmark for comparison. The method is presented in Algorithm 1. It starts with a single component mixture and iteratively splits (lines 4-7), deletes (lines 9-12), and merges (lines 14-18) components until no improvements in the message length is observed (line 11). This is a greedy approach, which is advantageous in that it is guaranteed to only accept perturbations that minimise the message length, but that guarantee comes with a computational cost.

Algorithm 1 The benchmark method [29].

Input: y
Output: ψ

current ← one component mixture
while TRUE **do**
    components ← current
    for $i \in K$ **do** ▷ Exhaustively split all components
        splits[i]← SPLIT(current,components[i])
    end for
    bestSplit←BEST(splits) ▷ Record the best split.
    if K>1 **then**
        for $i \in K$ **do** ▷ Exhaustively delete all components
            deletes[i]← DELETE(current,components[i])
        end for
        bestDelete←BEST(deletes) ▷ Record the best delete.
    end if
    for $i \in K$ **do** ▷ Exhaustively merge all components
        j← CLOSESTCOMPONENT(i)
        merges[i]← MERGE(merges)
    end for
    bestMerge←BEST(merges) ▷ Record the best merge.
    bestPerturbation ← BEST(bestSplit, bestDelete,bestMerge) ▷ Select the best perturbation.
    $\Delta I \leftarrow$ MESSAGELENGTH(bestPerturbation)-MESSAGELENGTH(current) ▷ Check for improvement.
    if $\Delta I < 0$ **then**
        current ← bestPerturbation
        CONTINUE
    end if
    BREAK
end while

5.2. The Message-Breaking Method

The simplest strategy we introduce here, *The Message-Breaking Method*, is a direct improvement upon the Benchmark Method that does not use predictions of the message length of future mixtures. Instead the algorithm begins with a $K = 1$ mixture and iteratively splits the component with the largest message length until the total message length no longer improves (Algorithm 2). This approach is related to the benchmark method in that we iteratively move from a $K = 1$ mixture, but we only split the component with the largest message length instead of splitting, merging, and deleting all components (and running EM at each operation).

We evaluate the message length of each component by rearranging Equation A.16 and multiplying the negative log-likelihood $-\log \mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$ by the $N \times K$

responsibility matrix $\mathcal{R}$ such that the message length of the kth component is

$$\begin{aligned} I_k &= -\sum_{n=1}^{N} \mathcal{R}_{nk} \log w_k f(\mathbf{y}_n|\boldsymbol{\mu}_k, \mathbf{C}_k) - \frac{(D+2)}{2} \log |C_k| \\ &\cdots + \left(\frac{D(D+3)}{4} - \frac{1}{2}\right) \log w_k + \left(1 - \frac{D}{2}\right) \log 2 \\ &\cdots + \frac{1}{K}\left[\log \Gamma(K) + \frac{1}{2}\left(\log Q\pi - Q \log 2\pi\right) + \frac{Q}{2} \log N\right] \end{aligned} \tag{20}$$

and the total message length remains the sum of the message lengths of the components

$$I(\mathbf{y}|\boldsymbol{\psi}, N, D) = \sum_{k=1}^{K} I_k \quad . \tag{21}$$

The component-wise calculation of the message length is performed as part of the expectation step of the expectation-maximisation algorithm, adding negligible computational overhead to the algorithm.

Algorithm 2 The Message-Breaking Method.

Input: $\mathbf{y}$ ▷ The $N \times D$ dimensional data.
Output: $\boldsymbol{\psi}$ ▷ The optimal mixture parameters.

```
current ← one component mixture
while TRUE do
    m ← ComponentMessageLength(current)
    i ← ArgMax(m)
    perturbation ← Split(current, components[i])
    ΔI ← MessageLength(perturbation) - MessageLength(current)
    if ΔI < 0 then                                  ▷ Check for improvement.
        current ← perturbation
        Continue
    end if
    Break
end while
```

When splitting the component with the largest message length, we split this component into two child components. Initially we set the means of the child mixtures as positive and negative eigenvalues along the principal eigenvector of the component covariance matrix [29]. A matrix of child responsibilities is then calculated based on the Euclidean distance between each datum and the child means (i.e., hard association), which we use to estimate the covariance matrices and weights of the child components. After the parameters of the child components are initialised, we run *partial* expectation-maximization to iteratively update the parameter estimates of the child components. By *partial*

expectation-maximization we refer to the process of keeping the parameters of all other components fixed, and only updating the estimates of the parameters of the child components [29]. After partial E-M has converged to the same specified tolerance given for the search, we integrate the child components with the rest of the mixture and run full E-M before the SPLIT operation (line 5 of Algorithm 2) is considered complete.

One advantage of the Message-Breaking Method is that the total number of E-M steps is relatively few because all partial E-M steps are performed from a near-optimal initialisation point. One possible disadvantage to this approach is that there may be mixtures where the SPLIT operation increases the message length by ϵ, but a subsequent SPLIT operation would *decrease* the total message length by $> \epsilon$. As a result, the algorithm would converge to a local minima, not the global one. In the code implementation that we release with this work, we have implemented a *Strict Message-Breaking Method* variant that allows the user to specify the number of successive K trials that increased the message length, before adopting the mixture with the shortest message length. By default the Strict Message-Breaking Method requires five successive mixtures where the message length did not improve.

5.3. Message-Jumping Method

The *Message-Jumping Method* seeks to find the mixture parameters $\boldsymbol{\psi}$ that produce the minimum message length over the class of finite mixtures $\boldsymbol{\mathcal{M}}$ in the fewest number of computations. This objective is shared by the *Message-Breaking Method* and the *Message-Jumping Method*, but they differ in their approach.

In the *Message-Breaking Method* we seek to minimize the total number of computations by using partial E-M to minimise the total number of (effective) E-M steps required to converge on a $K+1$ component mixture from a K component mixture. In the *Message-Jumping Method* we seek to minimise the total number of computations by optimizing a fewer number of mixtures. Instead of iteratively perturbing a mixture from say $K = 10$ to $K = 30$, we use predictions of the message length of untrialled mixtures and immediately 'jump' to a component (e.g., $K = 27$) that will provide the most improvement in our predictions of the minimum message length over *all* components. In doing so we seek a global message length minima with a fewer number of total E-M operations.

We describe the *Message-Jumping Method* in Algorithm 2. In this method we seek to predict the minimum message length for any number of K components, and then converge to the optimal mixture as quickly as possible. For this reason, we initialise many mixtures that are evenly separated in $\log_{10}$ space from $K = 1$ to $K = N/2$ (the maximum possible number of components) in order to quickly describe the minimum message length across all K. Specifically we introduce a search hyperparameter K_{inits} (defaults to 10) that sets the number of mixtures to initialise – and those initial K values.

We initialise each mixture using the K-means++ algorithm[**?**] and optimize the mixture using EM as described earlier. We trial each of the K_{init} mixtures in order of increasing K. If one of these mixtures fails to initialise or optimize,

this is likely because of zero entries in the diagonal of the covariance matrices. In practice this is typically because the estimate of K is much higher than K_{true}. In this situation we do not attempt to initialise any more of the K_{init} mixtures, and we being the search method.

The motivation for this initialisation process is to mimic the search strategy that a human might take: What is an order-of-magnitude estimate on the number of K? Or rather: what is a ballpark number on the maximum number of K that we should trial? After we have that estimate, and we have an estimate on the message length for a few sparsely-sampled numbers of components, we can make reasonable predictions on the global minimum.

We record the message length for the optimal mixture at each trialled K, and the model parameters for that mixture. With these quantities we predict the message length of untrialled mixtures for every K value from $K = 1$ to some number. The message length predicted from the optimised Gaussian processes provide an expectation value on the message length for an untrialled mixture, as well as a variance in that prediction (Figure **??**). Before selecting the next value of K to trial, we use the expectation value and associated variance to compute the acquisition function[**?**] (or the "expected improvement") at each value of K

$$\Psi = X \tag{22}$$

and select the next K value by choosing the K that will maximize the expected improvement.

After the next K value has been chosen, we initialise a new mixture with K components using the K-means++ algorithm[**?**] and run E-M until convergence. If the initialisation and optimization process fails, we reselect the next K value that will maximise the expected improvement. After the mixture has converged, we update our predictions for the message length of untrialled mixtures and choose the next K value to trial.

Algorithm 3 The Component-Jumping Method.

Input: $\mathbf{y}$, K_i, I_ϵ
Output: $\boldsymbol{\psi}$ ▷ The optimal mixture parameters.

s $\leftarrow (K_i)^{-1} \log_{10} N$ ▷ Initial step size in K.
v $\leftarrow [0, 1, ..., K_i]$
$K_{inits} \leftarrow$ Power(10, vs) ▷ Set initial K values to trial.
list K_t ▷ Create a list to save trialled K values.
for $k \in K_{inits}$ **do**
    K_t.append(k) ▷ Remember that we trialled this K value.
    try
        mix $\leftarrow$ KMeans^{++}(k) ▷ Initialise with K-means++.
        mix $\leftarrow$ ExpectationMaximization(mix)
    catch
        $K_{max} \leftarrow k$ ▷ Record the maximum K value.
        Break ▷ Do not trial any higher K values.
    finally
        mixtures[k] $\leftarrow$ mix
        states[k] $\leftarrow$ MessageLength(mix)
    end try
end for
converged $\leftarrow$ FALSE
while not converged **do**
    $K_{predict} \leftarrow [1, \cdots, K_{max}]$ ▷ Predict future message lengths.
    $I_{predict} \leftarrow$ PredictMessageLength($K_{predict}$,mixtures,states)
    $A \leftarrow$ Acquisition($K_{predict}, I_{predict}$) ▷ Calculate the expected improvement.
    $K_{next} \leftarrow$ ArgSort($I_{predict}$) ▷ Sort K by predicted message length.
    for $k \in K_{next}$ **do**
        if $k \in K_t$ **then**
            converged $\leftarrow$ TRUE
            Break
        else
            K_t.append(k) ▷ Remember that we trialled this K value.
            try
                mix $\leftarrow$ KMeans^{++}(k) ▷ Initialise with K-means++.
                mix $\leftarrow$ ExpectationMaximization(mix)
            catch
                Continue ▷ Move on to next K trial.
            finally
                mixtures[k] $\leftarrow$ mix
                states[k] $\leftarrow$ MessageLength(mix)
            end try
        end if
    end for
end while

6. Experimental Design

We conducted a set of experiments and real datasets to evaluate the proposed search algorithms.

6.1. Generated Datasets

We evaluated the following methods on all generated datasets: the benchmark method (Algorithm 1), the Message-Breaking Method (Algorithm 2), and X. The cost (in seconds) is shown as a function of the number of data points N and true number of clusters K in Figure **??**. Cost is approximately linear for MBM, but of O(3) for kasarpu. Not surprising because Kasarapu is a greedy search method, requiring at least 3*K operations (split, merge, delete) at each mixture.

We considered the search algorithm to be converged to the global minima if the calculated message length is shorter than the message length given the true mixture parameters. Something about when converged and not

Figure: predictions for future mixtures for one generated data set

Andy ▶*As the output of the algorithms depends on the starting seed, and some of the methods can be affected by randomness, we perform 50 runs for each search scheme and dataset. The different search schemes are validated using the Kolmogorov-Smirnov (KS) non-parametric test [32], which checks for a statistical significance in the difference between the performances of the algorithms. The 50 runs are submitted to the KS analysis with a null hypothesis of no difference between the performances of the different search methods.*◀

6.2. Galactic Archaeology data

We use detailed chemical abundances from the Apache Point Observatory Galactic Evolution Experiment (APOGEE), an experiment run by the Sloan Digital Sky Survey (SDSS) collaboration. We make use of SDSS Data Release 12, which includes high-resolution, high S/N infrared spectra for ∼163,000 stars collected by APOGEE. Each observation is an array of 8,575 pixels that have an associated rest-frame wavelength, and a measurement of the stellar flux that is normalised relative to the spectral energy distribution such that a flux value of 1 indicates no stellar absorption and a flux value near 0 implies strong stellar absorption. All else being equal, a stronger amount of absorption (lower normalised flux value) at a particular wavelength indicates that the star contains more of the chemical element that absorbs and emits light at that wavelength. For example, sodium emits and absorbs light at 1639.33 nm in vacuum conditions. All else being equal, a star with lower normalised flux at a rest-frame wavelength of 1639.33 nm indicates that the star has a higher abundance of sodium.

We make use of detailed chemical abundances measured for 96,648 APOGEE spectra[**?**]. This data set includes 15 measured chemical abundances for all stars (e.g., no missing data), with estimated uncertainties. All chemical abundances are reported as the logarithmic count of the number of atoms relative to hydrogen, with a correction to account for the Sun's chemical composition such that a

value of [Na/H] = 0 indicates a star has the same abundance of sodium relative to hydrogen as the Sun. To provide some context for the data, in Figure **??** we show the measured abundance of various elements relative to the measured abundance of [Fe/H] for all 96,648 stars. The Sun's abundance is marked.

The data set also contains other measured properties for each star. This includes the effective temperature and surface gravity, and whether they are likely associated with any gravitationally-bound star cluster. Most stars are not likely members of any known star cluster; most stars in this dataset are described as 'field' stars because they have no obvious association. We do not make use of any of this ancillary data (e.g., temperature, cluster association) during our search, but we do make use of it after the search is complete in order to evaluate the performance of our search.

7. Results

We recorded the wall clock time required by each search algorithm on generated data sets, which we show in Figure 2. Although wall-clock time metrics are subject to decisions made during implementation, all search and optimisation methods are implemented in the same way (using the same objective function), so the relative differences between methods is reliable. The *Message-Breaking Method* and *Message-Jumping Method* have approximately the same costs in wall time as a function of the number of clusters.

Costs of generated data sets. Switchover in time and we are faster

Convergence of generated data sets.

Expected complexity of [29] given the search setup

The *Message-Breaking Method* splits the component with the largest message length (Equation 20), whereas the benchmark method [29] tries to split, merge, and delete each component. After repeated trials on generated data sets, we found that the benchmark method [29] would most frequently chose to split the same component that the *Message-Breaking Method* would split (e.g., the component with the largest message length). We tracked the operation type (e.g., split, delete, merge) and which component was operated upon for X operations in Y generated data sets for the benchmark method [29] and the *Message-Breaking Method*.

The benchmark method chose to split in all X operations (e.g., no delete or merge operations). We also found that the benchmark method usually (X/X; X%) chose to split the same component that the *Message-Breaking Method* split: the component with the largest message length. The results are shown in Figure 1. Some X% of the time the component with the second largest message length was split by the benchmark method, X% of the time with the third-largest, and X% with the fourth-largest. Despite there being X operations with more than 4 components available to split, the benchmark method consistently chose one of four components with the largest message lengths. Despite the path differences taken between the benchmark method and the *Message-Breaking Method*, the optimised mixture parameters and message lengths found by each

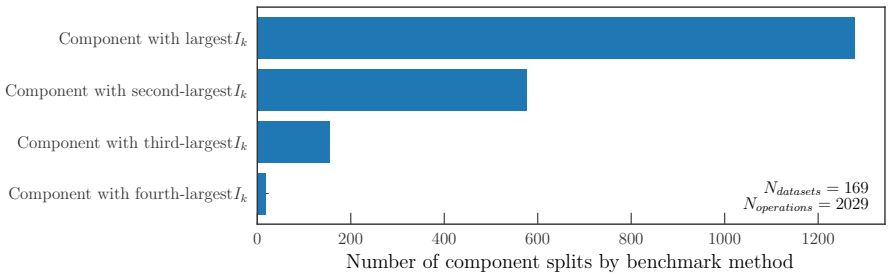

Figure 1: The number of split operations performed by the benchmark method [29] on the top four components (by message length) from 169 generated data sets. The *Message-Breaking Method* always splits the component with the largest message length. Although the benchmark method tries to split, merge, and delete all components, it most frequently decides to split the same component that the *Message-Breaking Method* splits – without trialling all operations. The benchmark method chose to split (instead of merge or delete) in all 2029 operations.

search algorithm were usually the same within the E-M convergence tolerance. This indicates that the *Message-Breaking Method* is just as effective as the benchmark method on generated data sets, and more efficient because it does not need to trial all possible operations on all possible components.

Figure: corner plot showing chemical abundance data

Figure: corner plot showing chemical abundance data, coloured by their responsibility

Figures showing the concentration of chemically-identified data

What do we find for the chemical abundance data

Recovering known star clusters that are still gravitationally bound?

Scientific insight from identifying groups in these data?

Estimate of the number of star-forming sites in the galaxy?

8. References

[1] K. Freeman, J. Bland-Hawthorn, The new galaxy: signatures of its formation, Annual Review of Astronomy and Astrophysics 40 (1) (2002) 487–537.

[2] E. Pancino, R. Carrera, E. Rossetti, C. Gallart, Chemical abundance analysis of the open clusters cr 110, ngc 2099 (m 37), ngc 2420, ngc 7789, and m 67 (ngc 2682), Astronomy & Astrophysics 511 (2010) A56.

[3] D. W. Hogg, A. R. Casey, M. Ness, H.-W. Rix, D. Foreman-Mackey, S. Hasselquist, A. Y. Q. Ho, J. A. Holtzman, S. R. Majewski, S. L. Martell, S. Mészáros, D. L. Nidever, M. Shetrone, Chemical Tagging Can Work: Identification of Stellar Phase-space Structures Purely by Chemical-abundance Similarity, The Astrophysical Journal 833 (2016) 262. doi:10.3847/1538-4357/833/2/262.

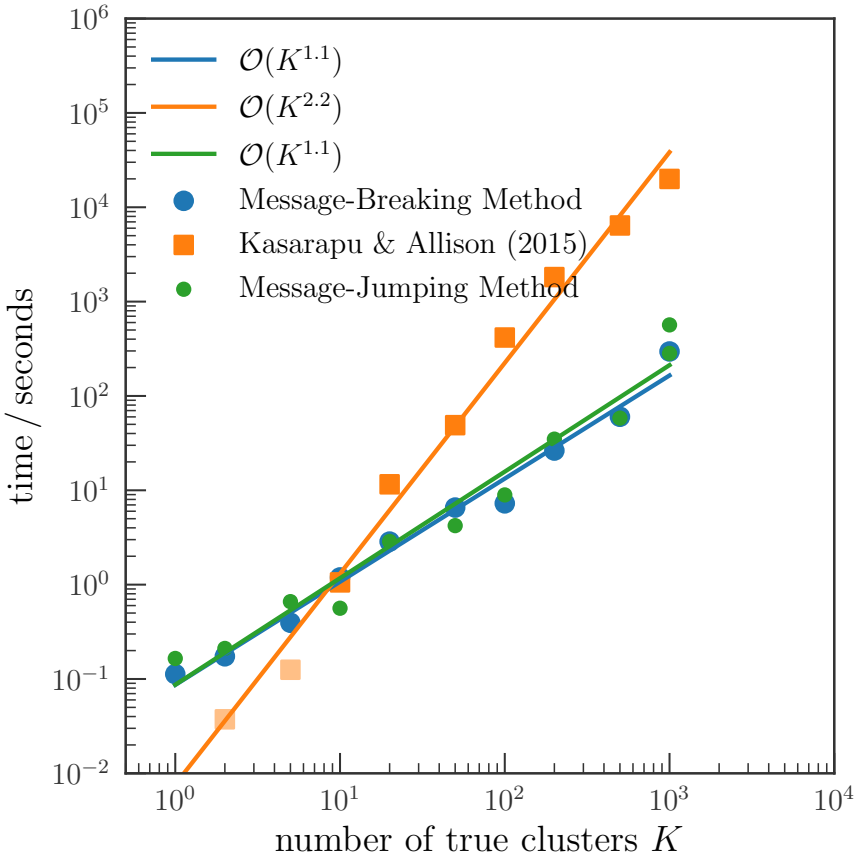

Figure 2: Elapsed wall clock time for trialled search strategies to converge on generated datasets as a function of the true number of clusters in the data. The approximate cost functions are shown.

[4] S. Buder, M. Asplund, L. Duong, J. Kos, K. Lind, M. K. Ness, S. Sharma, J. Bland-Hawthorn, A. R. Casey, G. M. De Silva, V. D'Orazi, K. C. Freeman, G. F. Lewis, J. Lin, S. L. Martell, K. J. Schlesinger, J. D. Simpson, D. B. Zucker, T. Zwitter, A. M. Amarsi, B. Anguiano, D. Carollo, L. Casagrande, K. Čotar, P. L. Cottrell, G. Da Costa, X. D. Gao, M. R. Hayden, J. Horner, M. J. Ireland, P. R. Kafle, U. Munari, D. M. Nataf, T. Nordlander, D. Stello, Y.-S. Ting, G. Traven, F. Watson, R. A. Wittenmyer, R. F. G. Wyse, D. Yong, J. C. Zinn, M. Žerjal, The GALAH Survey: second data release, The Monthly Notices of the Royal Astronomical Society 478 (2018) 4513–4552. doi:10.1093/mnras/sty1281.

[5] A. Mitschang, G. De Silva, D. Zucker, B. Anguiano, T. Bensby, S. Feltzing, Quantitative chemical tagging, stellar ages and the chemo-dynamical evolution of the galactic disc, Monthly Notices of the Royal Astronomical Society 438 (4) (2014) 2753–2764.

[6] H. Akaike, Determination of the number of factors by an extended maximum likelihood principle, Institute of Statistical Mathematics, 1971.

[7] S. E. Shimony, Finding maps for belief networks is np-hard, Artificial Intelligence 68 (2) (1994) 399–410.

[8] C. S. Wallace, D. M. Boulton, An information measure for classification, Computer Journal 11 (2) (1968) 185–194.

[9] C. S. Wallace, P. R. Freeman, Estimation and inference by compact coding, Journal of the Royal Statistical Society series B 49 (3) (1987) 240–252.

[10] C. S. Wallace, D. L. Dowe, Minimum message length and Kolmogorov complexity, Computer Journal 42 (4) (1999) 270–283.

[11] C. S. Wallace, Statistical and inductive inference by minimum message length, Springer Science & Business Media, 2005.

[12] M. Viswanathan, C. Wallace, D. L. Dowe, K. Korb, Finding outpoints in noisy binary sequencesa revised empirical evaluation, Advanced Topics in Artificial Intelligence (1999) 405–416.

[13] L. J. Fitzgibbon, D. L. Dowe, F. Vahid, Minimum message length autoregressive model order selection, in: Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on, IEEE, 2004, pp. 439–444.

[14] D. L. Dowe, S. Gardner, G. Oppy, Bayes not bust! why simplicity is no problem for bayesians, The British Journal for the Philosophy of Science 58 (4) (2007) 709–754.

[15] D. L. Dowe, Foreword re C. S. Wallace, Computer Journal 51 (5) (2008) 523 – 560, Christopher Stewart WALLACE (1933-2004) memorial special issue.

[16] D. L. Dowe, MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness, in: P. S. Bandyopadhyay and M. R. Forster (Ed.), Handbook of the Philosophy of Science - Volume 7: Philosophy of Statistics, Elsevier, 2011, pp. 901–982.

[17] D. L. Dowe, C. S. Wallace, Resolving the Neyman-Scott problem by Minimum Message Length, in: Proc. Computing Science and Statistics - 28th Symposium on the interface, Vol. 28, 1997, pp. 614–618.

[18] H. Akaike, A new look at the statistical model identification, IEEE transactions on automatic control 19 (6) (1974) 716–723.

[19] G. Schwarz, Estimating the dimension of a model, The annals of statistics 6 (2) (1978) 461–464.

[20] J. Rissanen, Modeling by shortest data description, Automatica 14 (5) (1978) 465–471.

[21] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the royal statistical society. Series B (methodological) (1977) 1–38.

[22] C. S. Wallace, D. L. Dowe, Intrinsic classification by MML - the Snob program, in: Proc. 7th Australian Joint Conf. on Artificial Intelligence, World Scientific, 1994, pp. 37–44.

[23] C. S. Wallace, D. L. Dowe, MML mixture modelling of multi-state, Poisson, von Mises circular and Gaussian distributions, Proc 28th Symp. on the Interface (1997) 608–613.

[24] C. S. Wallace, D. L. Dowe, MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions, Statistics and Computing 10 (2000) 73–83.

[25] J. J. Oliver, R. A. Baxter, C. S. Wallace, Unsupervised learning using mml, in: ICML, 1996, pp. 364–372.

[26] M. A. T. Figueiredo, A. K. Jain, Unsupervised learning of finite mixture models, IEEE Transactions on pattern analysis and machine intelligence 24 (3) (2002) 381–396.

[27] S. J. Roberts, D. Husmeier, I. Rezek, W. Penny, Bayesian approaches to gaussian mixture modeling, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (11) (1998) 1133–1142.

[28] C. Biernacki, G. Celeux, G. Govaert, Assessing a mixture model for clustering with the integrated completed likelihood, IEEE transactions on pattern analysis and machine intelligence 22 (7) (2000) 719–725.

[29] P. Kasarapu, L. Allison, Minimum message length estimation of mixtures of multivariate gaussian and von mises-fisher distributions, Machine Learning 100 (2-3) (2015) 333–378.

[30] C. E. Shannon, A mathematical theory of communication, Bell System Technical Journal 27 (3) (1948) 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.
URL https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[31] J. H. Conway, N. J. A. Sloane, On the voronoi regions of certain lattices, SIAM Journal on Algebraic Discrete Methods 5 (3) (1984) 294–305. doi:10.1137/0605031.
URL https://doi.org/10.1137/0605031

[32] A. N. Pettitt, M. A. Stephens, The kolmogorov-smirnov goodness-of-fit statistic with discrete and grouped data, Technometrics 19 (2) (1977) 205–210.

[33] D. M. Boulton, C. S. Wallace, Journal of Theoretical Biology 23 (1969) 269–278.

[34] L. Knorr-Held, Analysis of incomplete multivariate data. j. l. schafer, chapman & hall, london, 1997., Statistics in Medicine 19 (7) (2000) 1006–1008. doi:10.1002/(sici)1097-0258(20000415)19:7<1006::aid-sim384>3.0.co;2-t.
URL https://doi.org/10.1002/(sici)1097-0258(20000415)19:7<1006::aid-sim384>3.0.co;2-t

[35] P. S. Dwyer, Journal of the American Statistical Association 62 (1967) 607–625.

[36] J. R. M. . H. Neudecker, Matrix differential calculus with applications in statistics and econometrics, Wiley, New York, 1988.

Appendix A. The MML objective function

In the following subsections we provide expressions for the message length to encode a gaussian mixture model, and our prior beliefs on the parameters of that model, before describing our full objective function.

Appendix A.1. The number of mixture components (or groups) K

We assume a prior on the number of components K of $p(K) \propto 2^{-K}$ [11, p279, sec. 6.8.2] such that the length of the optimal lossless message to encode K is

$$I(K) = -\log p(K) = K \log 2 + \text{constant} \quad . \tag{A.1}$$

This expresses our prior belief that there ought to be fewer components.

Appendix A.2. The relative weight w_k of each component

We assume a uniform prior on the individual weights w_k. The weights $\mathbf{w}$ are drawn from a multinomial distribution, such that only $K-1$ weights need to be encoded because $\sum_{k=1}^{K} w_k = 1$. Following from Equation 5, here we describe how our estimate of the message length of the multinomial parameter $\mathbf{w}$ quantifies the trade-off between stating the parameters $\mathbf{w}$ imprecisely, so as to keep the first part of the message short and place higher prior probability in the relevant uncertainty region, and the goodness-of-fit to the data, which is typically better if we state the estimates of the parameters more precisely. By balancing this trade-off, the length of this component is equal (within a small constant) to the negative log of the product of the prior probabilities of each group ($p(\mathbf{w})$) divided by the square root of the expected Fisher information ($\sqrt{\mathcal{F}(\mathbf{w_k})}$), where the Fisher information matrix contains the *expected* second-order partial derivatives of the log-likelihood function. This describes how sharply the likelihood function peaks, and as a result it quantifies how precisely the parameters should be stated. In the case of uniform prior probabilities,

$$p(\mathbf{w}) = (K-1)! \quad , \tag{A.2}$$

the Fisher information is

$$\mathcal{F}(\mathbf{w}) = \frac{N^{K-1}}{\prod_{k=1}^{K} w_k} \quad , \tag{A.3}$$

which gives the message length of this part to be

$$\begin{aligned} I(\mathbf{w}) &= -\log\left(\frac{p(\mathbf{w})}{\sqrt{\kappa_K{}^K \mathcal{F}(\mathbf{w})}}\right) \\ I(\mathbf{w}) &= -\log(K-1)! + \frac{K}{2}\log\kappa_K + \frac{K-1}{2}\log N - \frac{1}{2}\sum_{k=1}^{K}\log w_k \\ I(\mathbf{w}) &= -\log\Gamma(K) + \frac{K}{2}\log\kappa_K + \frac{K-1}{2}\log N - \frac{1}{2}\sum_{k=1}^{K}\log w_k \end{aligned} \tag{A.4}$$

where κ_K is a lattice constant that varies between $1/12$ and $1/(2\pi e)$ [33].

Appendix A.3. The mean $\boldsymbol{\mu}_k$ and covariance matrix $\mathbf{C}_k$ for all K Gaussian components

In order to properly encode the means $\boldsymbol{\mu}$ and covariance matrices $\mathbf{C}$ for all K mixtures, we must encode both our prior belief on those parameters and the determinant of the expected Fisher information matrix. For the k-th mixture the joint message length is

$$\begin{aligned} I(\boldsymbol{\mu}_k, \mathbf{C}_k) &= -\log\left(\frac{p(\boldsymbol{\mu}_k, \mathbf{C}_k)}{\sqrt{|\mathcal{F}(\boldsymbol{\mu}_k, \mathbf{C}_k)|}}\right) \\ I(\boldsymbol{\mu}_k, \mathbf{C}_k) &= -\log p(\boldsymbol{\mu}_k, \mathbf{C}_k) + \frac{1}{2}\log|\mathcal{F}(\boldsymbol{\mu}_k, \mathbf{C}_k)| \end{aligned} \tag{A.5}$$

and for all mixtures:

$$I(\boldsymbol{\mu}, \mathbf{C}) = -\sum_{k=1}^{K} \log p\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right) + \frac{1}{2} \sum_{k=1}^{K} \log \left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right| \quad . \tag{A.6}$$

We use an improper uniform prior on the mean $\boldsymbol{\mu}$ for all mixtures such that $\mathcal{U}(\boldsymbol{\mu}) = (-\infty, +\infty)$. However, the MML principle requires proper priors. In practice we make this prior proper by assuming the bounds on $p\left(\boldsymbol{\mu}\right)$ are very large, but we do so without actually specifying the value of the bounds because they add only a constant value to the total message length. We adopt a conjugate inverted Wishart prior for the covariance matrix of an individual mixture [Section 5.2.3 of 34]:

$$p\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right) \propto \left|\mathbf{C}_k\right|^{\frac{D+1}{2}} \quad . \tag{A.7}$$

Following Eq. A.6, we require the logarithm of the determinant of the expected Fisher information matrix for $\boldsymbol{\mu}_k$ and $\mathbf{C}_k$ in order to encode the full message. In practice the determinant of the Fisher information is non-trivial to evaluate, and as such we approximate the determinant of the Fisher information $\left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right|$ as the product of $\left|\mathcal{F}\left(\boldsymbol{\mu}_k\right)\right|$ and $\left|\mathcal{F}\left(\mathbf{C}_k\right)\right|$ [25, 27]. In order to derive $\left|\mathcal{F}\left(\boldsymbol{\mu}_k\right)\right|$ we take the second derivative of the log-likelihood function $\mathcal{L}(\mathbf{y}|\boldsymbol{\theta})$, which yields:

$$\left|\mathcal{F}\left(\boldsymbol{\mu}_k\right)\right| = (N w_k)^D |\mathbf{C}_k|^{-1} \quad . \tag{A.8}$$

We make use of an analytical expression for $\left|\mathcal{F}\left(\mathbf{C}_k\right)\right|$ [35, 36]:

$$\left|\mathcal{F}\left(\mathbf{C}_k\right)\right| = (N w_k)^{\frac{D(D+1)}{2}} 2^{-D} |\mathbf{C}_k|^{-(D+1)} \quad . \tag{A.9}$$

Combining these expressions gives our approximation for the determinant of the expected Fisher information $\left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right|$:

$$\begin{aligned}
\left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right| &\approx \left|\mathcal{F}\left(\boldsymbol{\mu}_k\right)\right| \cdot \left|\mathcal{F}\left(\mathbf{C}_k\right)\right| \\
\left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right| &\approx (N w_k)^D |\mathbf{C}_k|^{-1} (N w_k)^{\frac{D(D+1)}{2}} 2^{-D} |\mathbf{C}_k|^{-(D+1)} \\
\left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right| &\approx (N w_k)^{\frac{D(D+3)}{2}} 2^{-D} |\mathbf{C}_k|^{-(D+2)} \quad .
\end{aligned} \tag{A.10}$$

Substituting Eqs. A.7 and A.10 into Eq. A.6 gives the message length to encode the component parameters $\boldsymbol{\mu}$ and $\mathbf{C}$ for all K mixtures:

$$\begin{aligned}
I(\boldsymbol{\mu}, \mathbf{C}) &= -\sum_{k=1}^{K} \log p\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right) + \frac{1}{2} \sum_{k=1}^{K} \log \left|\mathcal{F}\left(\boldsymbol{\mu}_k, \mathbf{C}_k\right)\right| \\
I(\boldsymbol{\mu}, \mathbf{C}) &= -\sum_{k=1}^{K} \log |\mathbf{C}_k|^{\frac{D+1}{2}} + \frac{1}{2} \sum_{k=1}^{K} \log \left[(N w_k)^{\frac{D(D+3)}{2}} 2^{-D} |\mathbf{C}_k|^{-(D+2)} \right] \\
I(\boldsymbol{\mu}, \mathbf{C}) &= \frac{D(D+3)}{4} \sum_{k=1}^{K} \log N w_k - \frac{2D+3}{2} \sum_{k=1}^{K} \log |\mathbf{C}_k| - \frac{KD}{2} \log 2
\end{aligned} \tag{A.11}$$

Appendix A.4. The data given the model

If we assume that the data have homoskedastic noise properties, then the precision of each measurement $\mathcal{E}$ relates the probability of a datum $p(y_n)$ can be related to the probability density of the datum, given the model $p(y_n) = \mathcal{E}^D p(y_i|\mathcal{M})$. In this work we adopt $\mathcal{E} = 0.01$, but we note that the value of $\mathcal{E}$ has no effect on our inferences. The message length of encoding a datum given the model is,

$$I(\mathbf{y}_n) = -\log p(\mathbf{y}_n) = -D\log\mathcal{E} - \log\sum_{k=1}^{K} w_k f_k(\mathbf{y}_n|\boldsymbol{\mu}_k, \mathbf{C}_k) \tag{A.12}$$

and for the entire data:

$$I(\mathbf{y}|\boldsymbol{\theta}) \quad = \quad -ND\log\mathcal{E} - \sum_{n=1}^{N}\log\sum_{k=1}^{K} w_k f_k(\mathbf{y}_n|\boldsymbol{\mu}_k, \mathbf{C}_k) \quad . \tag{A.13}$$

The final part of the message is to encode the lattice quantisation constant, which arises from the approximation to the (so-called) strict MML, where parameters are quantised into intervals in high dimensional space in order to be losslessly encoded. We use an approximation of the lattice constant (see Sections 5.1.12 and 3.3.4 of [11]) such that,

$$\log\kappa(Q) = \frac{\log Q\pi}{Q} - \log 2\pi - 1 \quad , \tag{A.14}$$

where Q is the total number of free parameters in the model. Each Gaussian component in the mixture requires D parameters to specify $\boldsymbol{\mu}$, and a covariance matrix with off-diagonal terms requires $\frac{1}{2}D(D+1)$ parameters. Because only $K-1$ weights need to be encoded, the total number of free parameters is

$$Q = K\left[\frac{D(D+3)}{2} + 1\right] - 1 \quad . \tag{A.15}$$

Appendix A.5. The complete objective function

Combining the expressions for the message length (or information content) of each component, we arrive at our objective function I to minimise across all model parameters (including the number of mixtures, K):

$$\begin{aligned} I(\mathbf{y}|\boldsymbol{\psi}, N, D) \quad = \quad & -\log\mathcal{L}(\mathbf{y}|\boldsymbol{\theta}) - \frac{(D+2)}{2}\sum_{k=1}^{K}\log|C_k| \\ & \cdots + \left(\frac{D(D+3)}{4} - \frac{1}{2}\right)\sum_{k=1}^{K}\log w_k + K\left(1 - \frac{D}{2}\right)\log 2 \\ & \cdots + \log\Gamma(K) + \frac{1}{2}\left(\log Q\pi - Q\log 2\pi\right) + \frac{Q}{2}\log N \end{aligned} \tag{A.16}$$

Appendix B. Reproducibility

This project was developed in a public `git` repository that is accessible at:

UNKNOWN

This repository includes a well-tested `Python 3.7` implementation of the algorithms described here, scripts to reproduce our results, and LaTeX to compile this manuscript. This research can be reproduced in full using the following commands in a modern terminal:

```
git clone https://github.com/andycasey/gmmmml.git gmmmml
cd gmmmml/
git checkout UNKNOWN
./reproduce
```

Reproducing these results will require at least X Gb of free disk space and Y hours of compute time, mostly because of the extensive algorithm comparisons performed.

Appendix C. Code implementation of search strategies

The astute reader will recognise that each search 'strategy' we have described is implemented in the `gmmmml` codebase as a set of 'policies' that govern different behaviour of the search:

- *Initialisation*: how many mixtures should be initialised, and how?
- *Prediction*: when should we make predictions about the message length of future mixtures, and how far ahead should we make predictions?
- *Movement*: *what* should the next K be?
- *Repartition*: *how* should we move to the next K? Should we repartition an existing mixture, initialise a new mixture, or should we make a probabilistic decision?
- *Convergence*: when should we stop searching?

As an example, the *Message-Breaking Method* can equivalently be described by a set of policies that: *initialise* a single mixture $K = 1$; makes no *predictions*; *moves* a single step towards the minimum message length; *repartitions* the closest existing mixture by splitting the component with the maximum message length; and describes *convergence* at the point when the message length of the current mixture is worse than the previous mixture. This policy-strategy factorisation allows for the trivial implementation of flexible search strategies with complex behaviour. We encourage the development of new search strategies.