Louis Sharrock Imperial College London

Automatic Image Segmentation and Cell Counting

1. Summary

A major use of machine learning in the biological sciences is for automatic image processing, for example distinguishing cells (and their components, such as the nuclei) from background material. In this project, we use the Gaussian mixture model and K-means algorithm to (a) obtain an automatic segmentation of the image, and (b) automatically count the number of cells in the image.

2. Preliminaries

2.1. The Data

The data consist of a single image, stored as a multidimensional array. The first 2 dimensions of this array correspond to the coordinates of each pixel, while the 3rd dimension represents the colour of each pixel using 3 numbers in the range $[0, 1]$. The 3 components of each colour vector correspond to the respective intensities of the red, green and blue primaries.

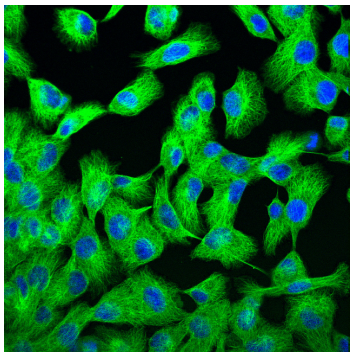


Figure 1: The original image.

3. Methods

We begin with a general summary of the K-means algorithm and the Gaussian mixture model. We also discuss their application to the task of image segmentation.

3.1. The K-Means Algorithm

Suppose that we have set of n observations $(\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathbb{R}^d$. The K-means algorithm attempts to partition these observations into $K \leq n$ clusters $\boldsymbol{\mathcal{C}} = (\mathcal{C}_1, \dots, \mathcal{C}_K)$ such that the within cluster sum

of squares (WCSS) is minimised. The WCSS simply refers to the sum of distances of points in each cluster, to the centroid or mean of that cluster. Thus, the algorithm aims to determine

$$\hat{\boldsymbol{\mathcal{C}}} = \arg\min_{\boldsymbol{\mathcal{C}}} \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{C}_j} ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \tag{1}$$

where $(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K)$ are the cluster means, defined as

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i \tag{2}$$

with $(N_1, \dots, N_K)$ denoting the total number of points allocated to each cluster. The K-means algorithm can be formulated as follows.

To begin, the K means are initialised, as $(\boldsymbol{\mu}_1^{(1)}, \dots, \boldsymbol{\mu}_K^{(1)})$. We will adopt the so-called Forgy method for initialisation, which randomly selects K observations from the data and sets these as the initial means.[1] The algorithm then proceeds in a two-step iterative fashion, alternating between the following two tasks:

(i) **Assignment**. Assign each data point to the cluster with the 'nearest' mean (or, more precisely, the cluster for which the Euclidean distance between that point and a cluster mean is minimised). This is precisely equivalent to assigning each observation to the cluster minimising the WCSS. Thus, at the t^{th} step, the j^{th} cluster has membership

$$\mathcal{C}_j^{(t)} = \left\{ \mathbf{x}_i : ||\mathbf{x}_i - \boldsymbol{\mu}_j^{(t)}||^2 \leq ||\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)}||^2 \ \forall 1 \leq k \leq K \right\} \tag{3}$$

Note that each point must be assigned to a unique cluster. A non-unique minimum distance is unlikely, but in the case of a tie we choose (arbitrarily) to assign $\mathbf{x}_i$ to the cluster with the smallest index j.

(ii) **Update**. Update the values of the cluster means to match the sample means of the data-points now in each cluster. That is,

$$\mu_j^{(t+1)} = \frac{1}{\sum_{i=1}^{K} \mathbb{1}(\mathbf{x}_i \in \mathcal{C}_j^{(t)})} \sum_{\mathbf{x}_i \in \mathcal{C}_j} \mathbf{x}_i \tag{4}$$

We remark that, since the arithmetic mean is a least-squares estimator, this also minimises the WCSS objective function.

These steps are repeated until the algorithm converges, in the precise sense that the cluster assignments of *all* data points no longer change.

Clearly, this algorithm can be applied for the task of image segmentation. In particular, suppose that our image has dimensions $n_1 \times n_2$. In a slight abuse of notation, we can represent the three dimensional array in which our image is stored as

$$\begin{pmatrix} \mathbf{y}_{11} & \cdots & \mathbf{y}_{1n_2} \\ \vdots & \ddots & \vdots \\ \mathbf{y}_{n_1 1} & \cdots & \mathbf{y}_{n_1 n_2} \end{pmatrix} \tag{5}$$

[1] Another common choice of initialisation is Random Partition, which first randomly assigns a cluster to each observation, and then assigns the initial means as the sample means of each cluster. Our choice of initialisation procedure is somewhat arbitrary, as regardless of choice of initial clusters, the K-means algorithm will always converge to a local optimum, but can never be guaranteed to globally converge. However, in general, the Random Partition method is preferred for algorithms such as the K-harmonic means and fuzzy k-means, while the Forgy method is preferred for expectation maximisation and standard K-means algorithms.

where $\mathbf{y}_{ij}$ corresponds to the three-dimensional intensity vector of the pixel with coordinates (i, j). We can rewrite this array as a set of $n_1 n_2$, observations in $\mathbb{R}^3$, namely,

$$(\mathbf{y}_{11}, \dots, \mathbf{y}_{n_1 1}, \mathbf{y}_{12}, \dots, \mathbf{y}_{n_1 2}, \dots, \mathbf{y}_{1n_2}, \dots, \mathbf{y}_{n_1 n_2}) \tag{6}$$

This set of data is precisely in the form required by the K-means procedure, and thus our algorithm can be applied to determine K cluster means, along with the membership of each cluster, for the specified image. To produce the desired segmentation, it remains to replace each data point $\mathbf{x}_i$ by its corresponding cluster mean $\boldsymbol{\mu}_j$, reformat the transformed data into its original form, and plot the results.

3.2. Gaussian Mixture Model

Suppose, once more, that we have a set of n observations $\mathcal{D} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$. We assume that the data is drawn from some underlying Gaussian mixture model

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} \pi_m p(\mathbf{x}|\boldsymbol{\theta}_m) \tag{7}$$

where $p(\boldsymbol{x}|\boldsymbol{\theta}_m) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \Sigma_m)$. We are then interested in determining the set of parameters $\boldsymbol{\theta} = (\pi_1, \dots, \pi_M, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)$, where $\boldsymbol{\theta}_m = (\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ denotes the mean and covariance of the m^{th} Gaussian in our mixture model, and π_m denotes the probability of a given data point being generated by that particular Gaussian.[2]

We generally do not have access to labels which denote which mixture component generated each of the data points. We thus introduce (latent) indicator variables $\mathbf{z}_i = (z_{1i}, \dots, z_{Mi})$ where $z_{mi} = 1$ denotes that data point $\mathbf{x}_i$ was drawn from the m^{th} component of our mixture model. Writing

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n) \quad , \quad \mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_n) \tag{8}$$

we then wish to determine the set of parameters such that the likelihood of the complete data, consisting of $(\mathbf{X}, \mathbf{Z})$, is maximised. For this purpose, we can utilise the Expectation-Maximisation (EM) algorithm. This is formulated as follows.

To begin, the parameters are initialised. In this instance, we initialise the means $(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M)$ as M randomly selected data points,[3] the covariances $(\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_M)$ as the sample covariances of M randomly selected samples of our data of size 100, and assign $p(m) = \frac{1}{M}$. The algorithm then iteratively alternates between two main steps. In the interest of brevity, we have chosen to omit details of the derivation of the following formulae.

(i) **Expectation**. Compute the probability density of the latent variables, $p(\mathbf{Z}|\mathbf{X})$, by maximising the likelihood of the data with respect to $p(m|\mathbf{x}_i)$, which denotes the probability that $z_{mi} = 1$ for any $i \in \{1, \dots, n\}$.[4] This is found to be given precisely by the posterior distribution over all mixture components m that generated the data $\mathbf{x}_i$:

$$p(m|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|\boldsymbol{\theta}_m)p(m)}{\sum_{m'=1}^{M} p(\mathbf{x}_i|\boldsymbol{\theta}_{m'})p(m')} \tag{9}$$

where $p(m)$ denotes the probability that $z_{mi} = 1$ for any $i \in \{1, \dots, n\}$.

[2] We remark that these parameters must satisfy the constraint $\sum_{m=1}^{M} \pi_m = 1$, since the probabilities of all mixture components must necessarily sum to unity.

[3] For the purpose of comparison, we also run the algorithm initialising the means $(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M)$ as the sample means of M randomly selected samples of the data of size 100, as for the covariances.

[4] More precisely, at this step we maximise the lower bound of the log-likelihood.

(ii) **Maximisation**. Maximise the likelihood with respect to the component parameters $\boldsymbol{\theta}_m$, and update the corresponding estimates accordingly. The new parameters are given explicitly by

$$\hat{\boldsymbol{\mu}}_m = \frac{\sum_{i=1}^n p(m|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^n p(m|\mathbf{x}_i)} \tag{10}$$

$$\hat{\boldsymbol{\Sigma}}_m = \frac{\sum_{i=1}^n p(m|\mathbf{x}_i)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_m)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_m)^T}{\sum_{i=1}^n p(m|\mathbf{x}_i)} \tag{11}$$

$$p(m) = \frac{1}{n}\sum_{i=1}^n p(m|\mathbf{x}_i) \tag{12}$$

We iterate these steps until a chosen convergence criterion is satisfied. Our particular convergence criterion is formulated in terms of the log-likelihood, namely,

$$\log l(\boldsymbol{\theta}) = \sum_{i=1}^n \log\left[p(\mathbf{x}_i|\boldsymbol{\theta})\right] = \sum_{i=1}^n \left[\log \sum_{m=1}^M \pi_m p(\mathbf{x}_i|z_m, \boldsymbol{\theta}_m)\right] \tag{13}$$

T Thus, the algorithm is terminated when the absolute change in the log-likelihood between two iterations is less than some specified tolerance ε.

It remains to consider how this algorithm can be applied in the context of image segmentation. We begin by specifying M (the number of mixture components) and ε (the tolerance level to determine convergence). The EM algorithm can then be applied to the image data to estimate $\{\boldsymbol{\mu}_m\}_{m=1}^m$ and $\{\Sigma_m\}_{m=1}^M$, as well as $\{p(m|\mathbf{x}_i)\}_{m=1}^M$.[5] For each $\mathbf{x}_i$, we then assign the mean $\boldsymbol{\mu}_m$ intensity vector corresponding to the mixture component for which $p(m|\mathbf{x}_i)$ is maximised. The resulting plot is precisely the required image segmentation.

4. Automatic Image Segmentation

4.1. Data Exploration

Initial inspection of the image indicates the prevalence of three main distinct colours: black, green and blue (Figure 1). In particular, as represented in this image, cells appear to have relatively well defined blue centres, surrounded by more ambiguous green region, with the 'background' coloured black. In this context, it is reasonable to suggest that $K = 3$ may represent a good starting point for our investigations.

4.2. Results

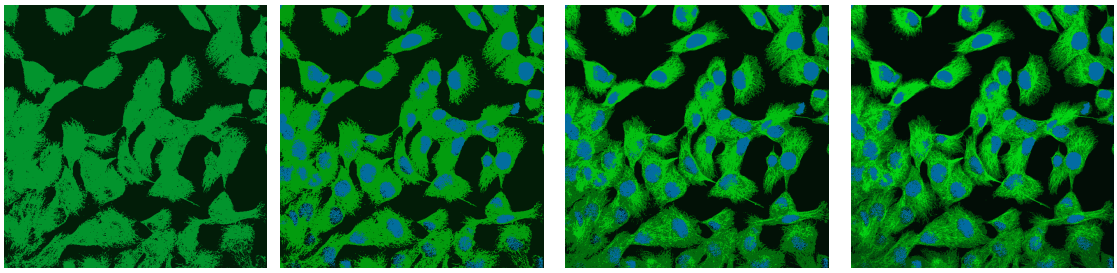
4.2.1 K-Means Algorithm

We first consider the output from the K-means algorithm. In particular, the following observations are pertinent:

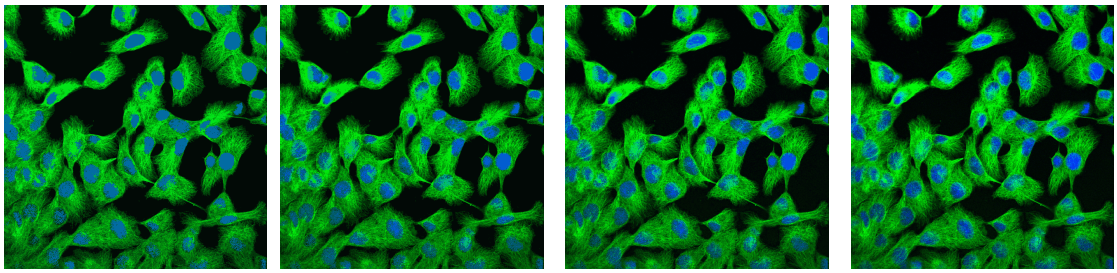
- The likeness of the segmented image and the original image increase as a function of K, as it to be expected.
- By the maximum considered value of $K = 50$, the segmented image is qualitatively almost indistinguishable from the original image .

[5] We assume that the data has been formatted as in equation (6).

- As predicted on the basis of our preliminary analysis, the results obtained in the case $K = 3$ provides a segmentation which encodes the 'main features' of the original image. In particular, it is essentially now possible to identify the position and count of cells on the basis of this image, by associating cells with cluster of pixels assigned to the 'blue' centroid vector (and surrounded by the pixels assigned to the 'green' centroid vector), and the image background with pixels assigned to the 'black' centroid vector.



(a) $K = 2$. (b) $K = 3$. (c) $K = 4$. (d) $K = 5$.



(e) $K = 7$. (f) $K = 10$. (g) $K = 20$. (h) $K = 50$.

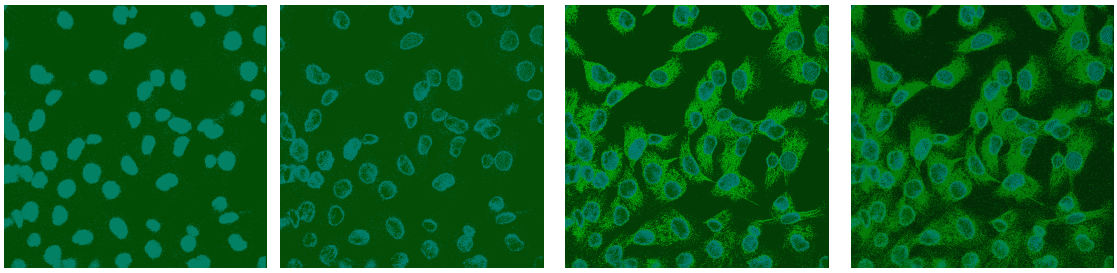
Figure 2: Segmented images produced using the K-means algorithm, for several values of K.

4.2.2 Gaussian Mixture Model

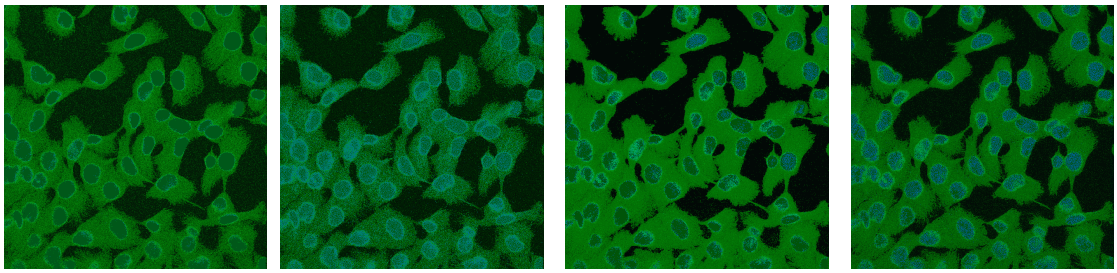
We now turn our attention to the Gaussian mixture model. In this case, we make the following observations:

- The output segmented images provide rather less resemblance to the original image, even as the number of components M increases.
- The mean intensity vectors assigned to pixels do not appear to closely correspond to the colours appearing most frequently in the original image, and, in particular, are somewhat less distinct.[6]
- The assignment of pixels is such that clusters are formed in a more 'jagged' fashion than in the original image. This is particularly evident for $M = 2$, whereby the image is partitioned clearly into clusters of pixels which could be seen to represent the nuclei of the cells in the image. As M increases, this observation remains valid: in general, the Gaussian mixture model does not capture the green 'strands' which appear to overlay the blue regions in the original image, but returns clusters of pixels with identical intensity vectors for these regions.

[6]Consider, for example, $K = 3$. It is not clear without close inspection of the output image that there are indeed three different intensity vectors assigned to pixels.



(a) $M = 2$. (b) $M = 3$. (c) $M = 4$. (d) $M = 5$.



(e) $M = 7$. (f) $M = 10$. (g) $M = 20$. (h) $M = 50$.

Figure 3: Segmented images produced using the Gaussian mixture model (and EM algorithm), for several values of M.

To conclude this commentary, we remark that the output from the Gaussian mixture model is highly dependent on initial starting conditions (results omitted). This is in contrast to the K-means algorithm.

4.3. Additional Remarks

Our previous observations, and in particular the stark contrast between the output from the two algorithms, are most readily explained with respect to the underlying models involved. In particular, K-means performs so-called 'hard clustering', in comparison to the 'soft-clustering' performed by GMM. As such, we expect the centroid vectors assigned by the K-means algorithm to correspond very closely to the colours which dominate the original image. Conversely, the GMM only assigns at each stage probabilities of belonging to each cluster, rather than a performing a hard assignment, which allows for much greater variation in the final colours assigned to pixels, and a much greater dependence on the initial conditions.

5. Automatic Cell Counting

5.1. Outline

In order to automate the counting of the number of cells in the image, we can use the output from the automatic image segmentation in another Gaussian mixture model. Broadly speaking, this automated count is achieved by determining the 'optimal' number of components in a Gaussian mixture model for the coordinates of pixels identified as representing each of the cells (as output by our initial algorithm). We provide more details of this approach below.

5.2. Methodology

We begin by recalling that applying the K-means algorithm to the image data with $K = 3$ results in three centroid vectors corresponding, at least qualitatively, to the colours black, green and blue. The distinct cells, or nuclei, are represented by clusters of pixels assigned to the 'blue' cluster mean. That is, more precisely, to the cluster mean with the greatest value in its third dimension among the three output cluster means. Let us suppose that p pixels in our original image are assigned to the 'blue' cluster mean. The data-set of interest - that is, the data set to which we will apply another Gaussian mixture model - is then given by $\mathcal{D} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$, where $\mathbf{x}_i \in \mathcal{D}$ now denote the two-dimensional coordinate vectors of each of these pixels.

It remains to detail how the application of the Gaussian mixture model to this data can automate the process of counting cells. In fact, it is by fitting multiple such models, for different numbers of mixture components M, that the required count is determined. In particular, the (estimated) number of cells in the image corresponds to the value of M, say M_{count}, for which the corresponding mixture model achieves optimality in some well defined sense (with respect to a set of mixture models fit using some reasonable range of values of m).

An obvious candidate for this optimality criterion may appear to be the maximisation of the likelihood function. However, this quantity is strictly increasing as a function of M, and thus unsuitable. In fact, several different criteria are available. In this case, we consider minimisation of the Bayesian Information Criterion (BIC). This is defined as

$$\text{BIC} = k \ln(n) - 2 \log(L) \tag{14}$$

where k denotes the number of free parameters in the model, n the number of observations, and L the (maximised) likelihood.[7,8] For a Gaussian mixture model with M components, and observations $\mathbf{x}_i \in \mathbb{R}^d$, the number of free parameters is determined as follows. For each mixture component, we are required to estimate a mean vector $\boldsymbol{\mu}_m \in \mathbb{R}^d$ (d free parameters), a symmetric covariance matrix $\Sigma_m \in M_{d\times d}$ ($\frac{d(d+1)}{2}$ free parameters), and a 'mixture' probability, π_m (1 free parameter). We thus have a total of

$$k = M\left[d + \frac{d(d+1)}{2} + 1\right] - 1 = M\left[\frac{(d+2)(d+1)}{2}\right] - 1 \tag{15}$$

free parameters, where the minus one corresponds to the constraint that the mixture probabilities must sum to 1.

5.3. Results

The number of cells in the image is determined by our automated counting routine as 64 (Figure 4). This corresponds very closely to the 'correct' number of cells, obtained via a manual count. The particular value output by our algorithm should, however, be interpreted with a degree of caution. Indeed, it is evident from Figure 4 that neither of the optimality criteria are clearly minimised by one particular value of M.

In this context, it is expected that the particular output value may be susceptible to small changes in the initial parameters, including the randomisation scheme used to determine starting values for

[7] This quantity is closely related to the Akaike Information Criterion (AIC), defined as $\text{AIC} = 2k - 2\ln(L)$, which is also widely used.

[8] Our choice to use the BIC rather than the AIC is based largely on empirical results. In particular, fitting Gaussian mixture models to our data for a range of values of M, with prior knowledge of the 'correct' number of cells (which is approximately in the range 60-70), we observe that minimisation of the BIC occurs in the 'correct' range, while the AIC tends to be minimised for slightly higher values of M. This is largely unsurprising, given the BIC imposes a harsher penalty on the number of free parameters (i.e. the number of mixture components) than the AIC.

the GMM. This being said, there is a relatively clear 'range' of M for which the BIC is 'minimised'. On this basis, it is certainly reasonable to conclude on the basis of our automatic counting scheme that the number of cells in our image is somewhere in the range $[55, 85]$.

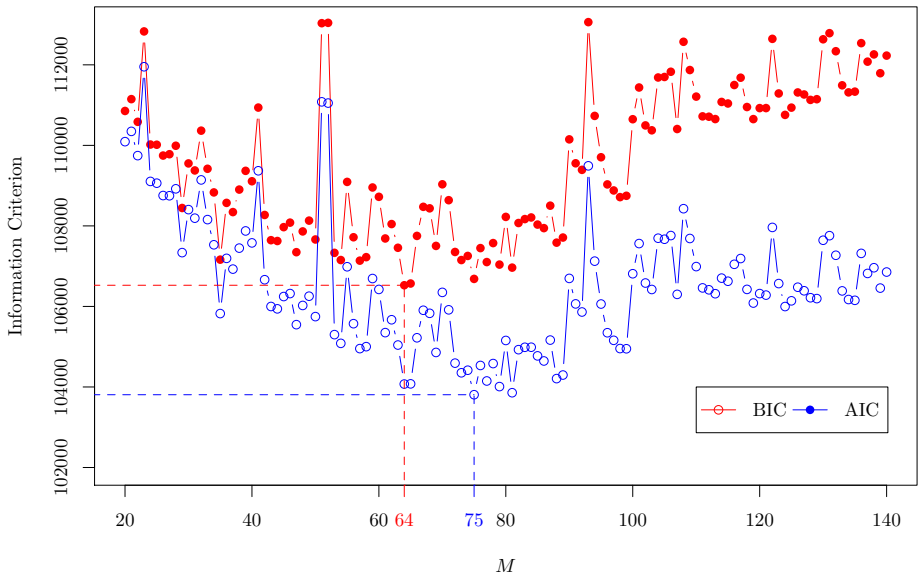


Figure 4: Plots of the BIC and AIC as functions of the number of mixture components M. The minimum of each information criterion (annotated) corresponds to an approximate value for the automated cell count.