# ECSE626 Statistical Methods in Computer Vision Assignment #2

Andrei Chubarau, 260581375, andrei.chubarau@mail.mcgill.ca

March 20, 2018

## 1 Part I: Gaussian Mixture Models

Image segmentation is implemented via unsupervised learning based on the Expectation Maximization (EM) algorithm using Gaussian Mixture Models (GMM). All code is done in Python, using Numpy [4] and OpenCV [1] packages. Note that all segmentation was performed on full resolution input images (no downscaling).

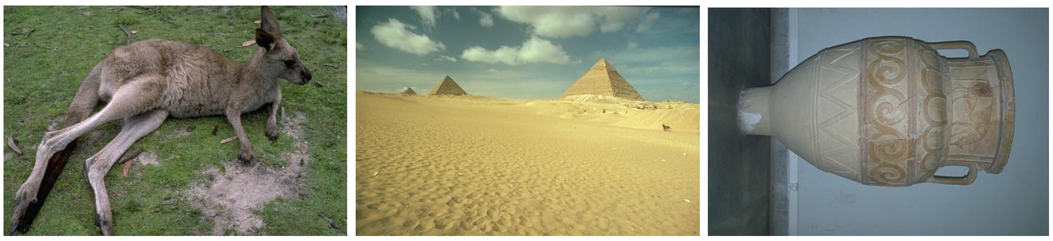
Firstly, raw images are converted into corresponding feature vectors as described in the assignment instructions. Three feature spaces are explored: the model can use Intensity, RGB or L*a*b features. The EM algorithm then maximizes the log likelihood of the probabilities in an iterative process to compute the segmentation labels. Convergence is detected via thresholding the amount of change to the log likelihood over subsequent iterations; a small enough change indicates that convergence has been achieved.

For computation of features, the following components were implemented. Color intensity $I$ of an RGB pixel is given by its luminance computed as $I = 0.2126R+0.7152G+0.0722B$. L*a*b color space conversion from RGB space was implemented by me as per [2].

Note that my implementation computes log likelihoods at the beginning of each EM iteration, and not at the end, i.e. it is computed before updates to means and covariance matrices of the Gaussians. This does not affect the correctness of the algorithm; it is an optimization (mostly for the sake of convenience and to avoid repetition) to avoid recomputing probabilities after updating Gaussian model parameters.

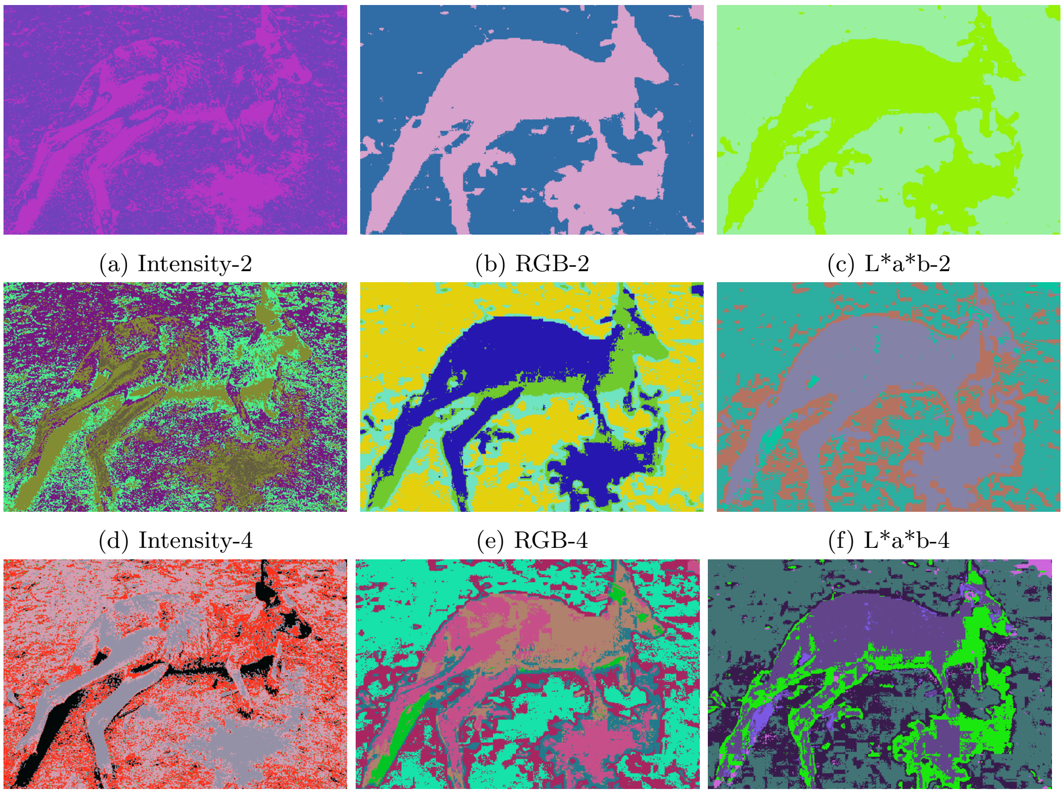
Finally, segmentation results are displayed where each region is given a randomized color to be distinguishable; some results for 2, 4, and 6 Gaussians are given in Figure 2.

The results show that RGB and L*a*b feature spaces offer significantly superior results to that of Intensity features. This is explained by the fact that these spaces offer more information for the clustering, i.e. one channel for intensity compared to three for RGB. Note that RGB and L*a*b seem to produce very similar results, where the winner is difficult to judge.



(a) 69020.jpg (b) 260058.jpg (c) 227092.jpg

Figure 1: Original images to be used for segmentation

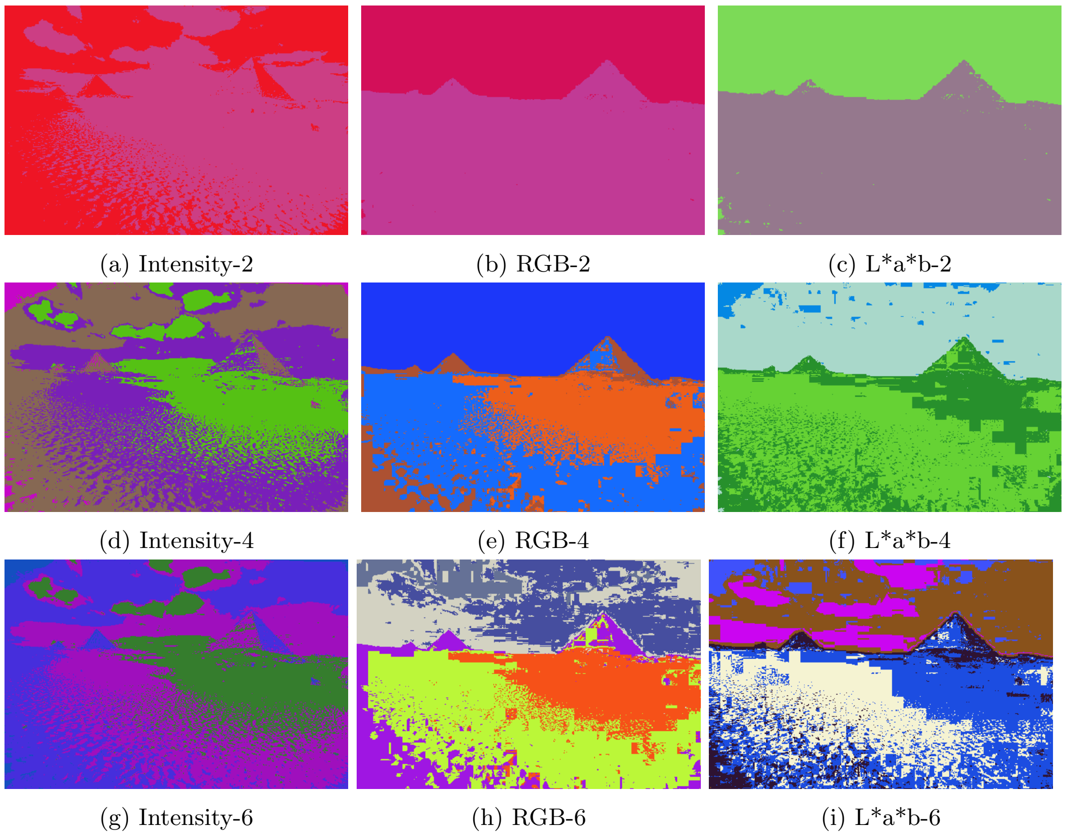


(a) Intensity-2 (b) RGB-2 (c) L*a*b-2

(d) Intensity-4 (e) RGB-4 (f) L*a*b-4

(g) Intensity-6 (h) RGB-6 (i) L*a*b-6

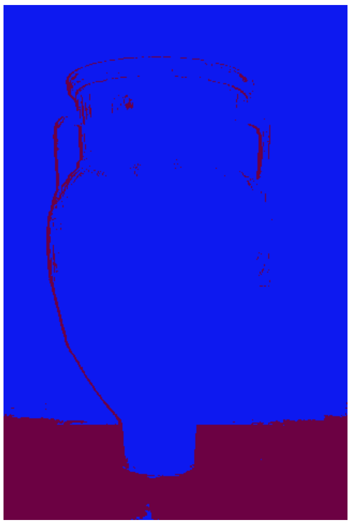
Figure 2: Segmentation results for image 69020.jpg using different number of Gaussians (2, 4, and 6) and three feature spaces (Intensity, RGB, L*a*b).

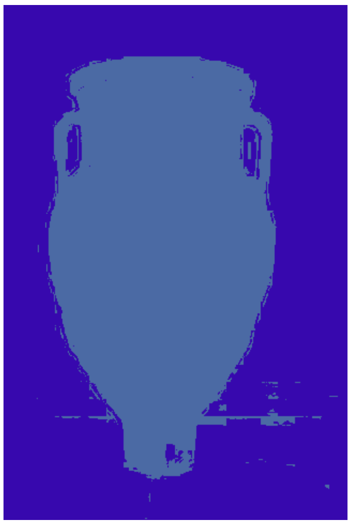

(a) Intensity-2 (b) RGB-2 (c) L*a*b-2

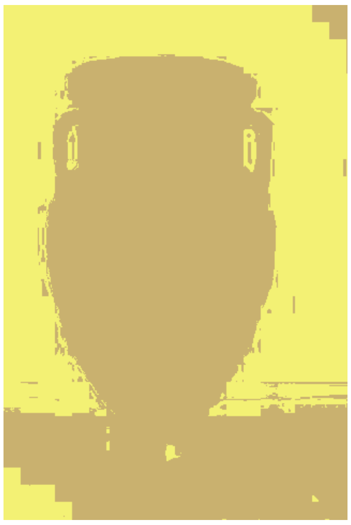
(d) Intensity-4 (e) RGB-4 (f) L*a*b-4

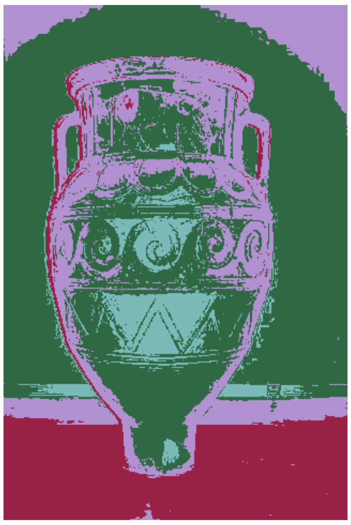
(g) Intensity-6 (h) RGB-6 (i) L*a*b-6

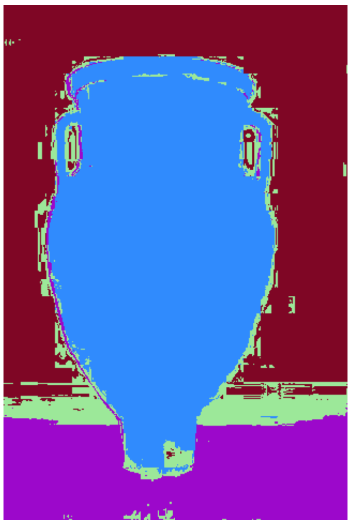
Figure 3: Segmentation results for image 260058.jpg using different number of Gaussians (2, 4, and 6) and three feature spaces (Intensity, RGB, L*a*b).

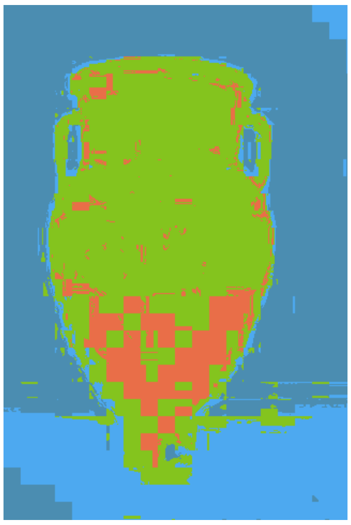

(a) Intensity-2

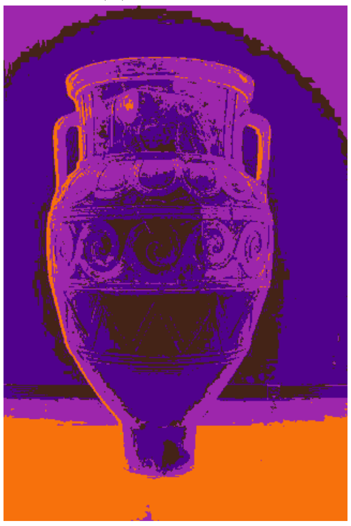

(b) RGB-2

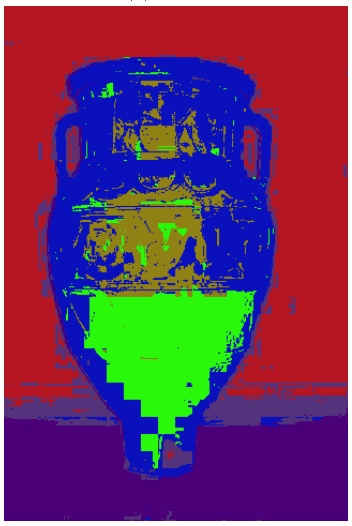

(c) L*a*b-2


(d) Intensity-4


(e) RGB-4


(f) L*a*b-4


(g) Intensity-6


(h) RGB-6

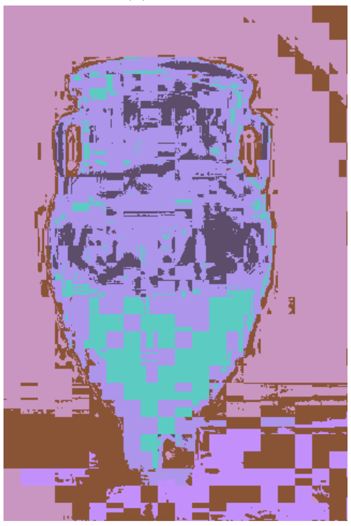

(i) L*a*b-6

Figure 4: Segmentation results for image 227092.jpg using different number of Gaussians (2, 4, and 6) and three feature spaces (Intensity, RGB, L*a*b).

## 2 Part II: Model Selection

#### 2.0.1 Cross Validation and Bayesian Information Criterion

Cross-validation generally refers to the process of model validation in the context of assessing how well a given statistical analysis will generalize to an independent data set.

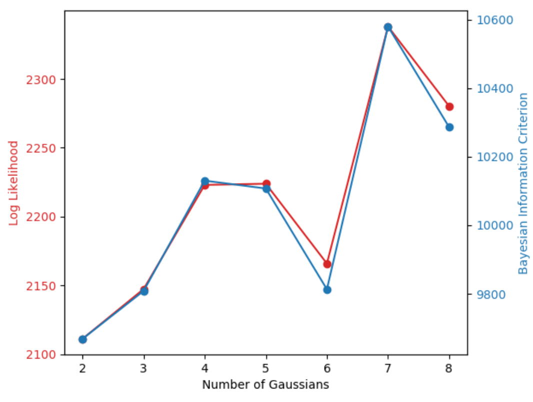
In my implementation, 5-fold cross-validation is done by dividing the originally N-long feature vector into five parts. Five training sessions are then carried out. Each time, the data is separated into a training and a testing set; the training set consists of four parts out of the five available, and the fifth part is left out for testing. Thus, the GMM parameters are computed using the training set, whereas validation accuracy is computed on the held-out validation set. This gives five log likelihood and BIC values, for which a final average can be computed. The results are shown in in Figure 5. Note that this procedure was carried out for each of the three feature spaces, namely using Intensity, RGB, and Lab features.

In theory, higher maximum likelihood and lower BIC value (or higher -BIC value) provide better segmentations. Both used metrics are quite similar, the only difference essentially being that BIC incorporates a penalty on the complexity of the model. The influence of the penalty is most observable when more Gaussians are used.

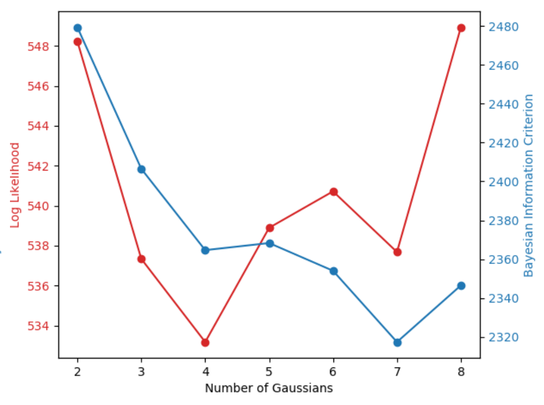
According to my results, the optimal number of Gaussians is as follows: i) for Intensity, 2 for L and BIC, ii) for RGB, 5 for L and 3 for BIC, iii) for L*a*b, 4 for L and BIC. The best parameters are given when the validation "accuracy" starts to fall off, while training accuracy still increases. This is the point where the models begin to over-fit the training data, as displayed by sudden decrease in the validation accuracy.

Having analyzed the results of running multiple sessions of cross-validation for both log likelihood and BIC values, it seems quite apparent that in the current use case, both methods are rather inconsistent. First of all, there is very high variance between runs; the plots are quite different every time cross-validation is run. While this is somewhat expected due rather small amounts of data and randomization of the training/validation sets, it is nonetheless a worrisome concern. Inconsistencies are mostly observable when using a feature space with low dimension; most inconsistent results appear for Intensity, slightly less so for L*a*b, and a lot less so for RGB space. This perhaps can be fixed by increasing the number of times cross-validation is run, but this was not done due to higher computational cost (currently, cross-validation using Intensity features takes about 2 minutes, while L*a*b and RGB go on for about 5-10 minutes).

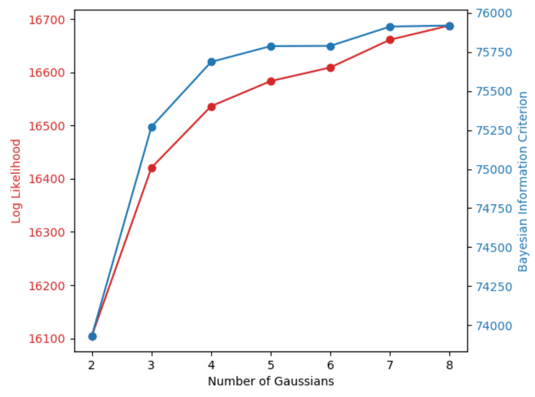
For all used features, training accuracy suggests that higher count of Gaussians is preferable. A case can be made to argue that this is because fitting more Gaussians would minimize error and thus maximize the likelihood; this is essentially the bane of over-fitting. Using BIC allows us to somewhat compensate for this effect; BIC penalizes model complexity incurred through using higher number of Gaussians thus biasing our solution towards a model where a trade-off between over fitting and model complexity allows us to draw a better conclusion regarding the optimal number of clusters. Nevertheless, both likelihood and BIC metrics produce quite similar cross-validation results.



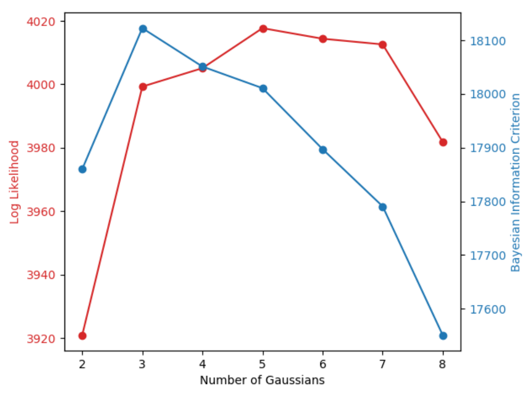
(a) Intensity, Training



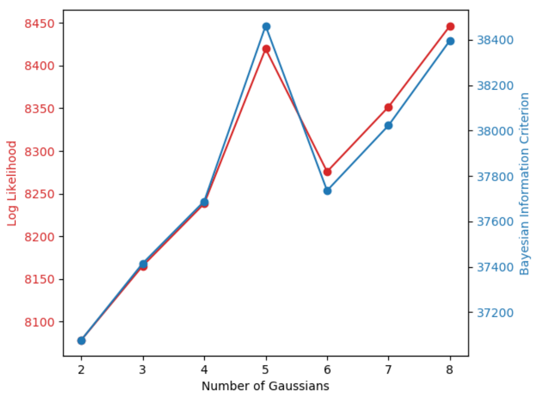
(b) Intensity, Testing



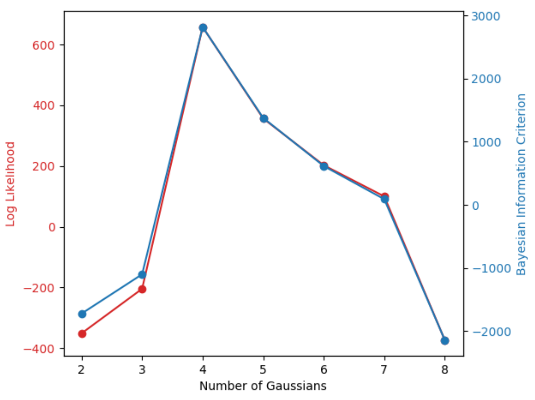
(c) RGB, Training



(d) RGB, Testing



(e) L*a*b, Testing



(f) L*a*b, Testing

Figure 5: Maximum likelihood and Bayesian Information Criterion values as function of number of Gaussians for different feature types for training and validation sets. Note that the shown BIC value had its sign flipped (original BIC values are largely negative), for the sake of convenience.

## References

[1] Itseez, “Open source computer vision library.” https://github.com/itseez/ opencv, 2015.

[2] Itseez, “OpenCV Color Conversions.”
https://docs.opencv.org/3.3.0/de/d25/imgproc_color_conversions.html. opencv, 2015.

[3] E. Jones, T. Oliphant, P. Peterson, et al., “SciPy: Open source scientific tools for Python,” 2001–.

[4] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, “The numpy array: A structure for efficient numerical computation,” Computing in Science and Engg., vol. 13, pp. 22–30, Mar. 2011.