EECE5644: Assignment 4

SVM, MLP, and GMM based Image Segmentation

Student: Atharva Prashant Kale

NUID: 002442878

Email: kale.ath@northeastern.edu

1. Introduction

This assignment studies two machine learning tasks:
1. Nonlinear binary classification using Support Vector Machine with an RBF kernel and a Multilayer Perceptron.
2. Unsupervised image segmentation using a Gaussian Mixture Model with model order selection using cross validation.

The synthetic dataset in Question 1 has two concentric circular classes with Gaussian noise, which forces nonlinear decision boundaries. Question 2 performs pixel level segmentation using a GMM.

2. Question 1: SVM and MLP on Concentric Rings

2.1 Data Generation

Each data point is created as follows.

θ is sampled from a uniform distribution:
$\theta \sim \text{Uniform}[-\pi, \pi]$

Noise is sampled from a 2D isotropic Gaussian:
$n \sim \text{Normal}(0, \sigma^2 I_2)$

Class radii:
$r_- = 2$ for class -1
$r_+ = 4$ for class $+1$

Clean point on ring:
$u = [\ r_l \cos(\theta), r_l \sin(\theta)\ ]$

Final noisy sample:
$x = u + n$

Label:
$l = -1$ for inner ring
$l = +1$ for outer ring

Training set size: 1000 samples
Test set size: 10000 samples
Both contain equal positive and negative samples and are shuffled.
The data is not linearly separable because the true boundary is circular. This motivates nonlinear classification.

2.2 Support Vector Machine with RBF Kernel

The SVM decision function is:

$$f(x) = \Sigma_i \alpha_i y_i K(x_i, x) + b$$

The RBF kernel is:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

Hyperparameters:

C controls the penalty for misclassification

γ controls the kernel width (smaller γ gives smoother boundaries)

Grid used:

$C \in \{0.1, 1, 10, 100\}$

$\gamma \in \{0.01, 0.1, 1, 10\}$

Before training, every feature is standardized:

$$\hat{x} = (x - \mu) / \sigma$$

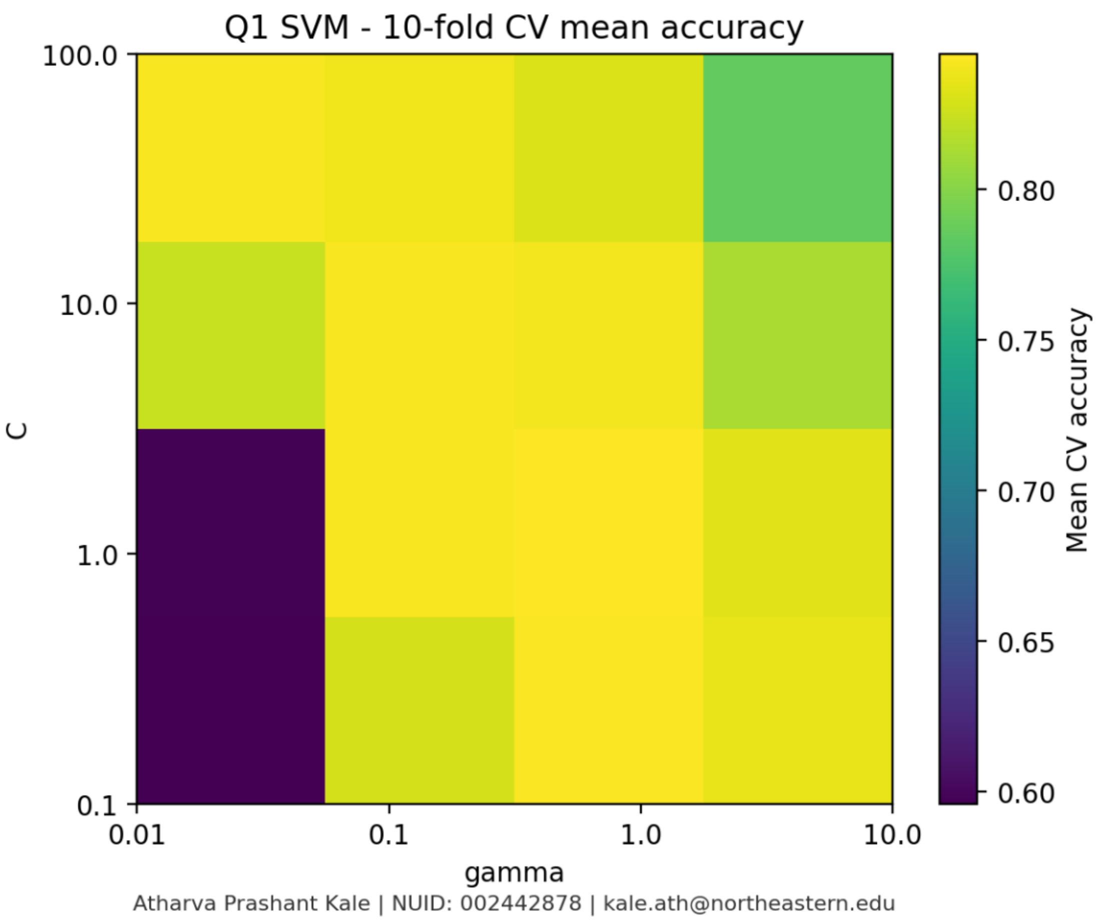

Figure 1: Q1 SVM ten fold cross validation mean accuracy heatmap for the grid $C \in \{0.1, 1, 10, 100\}$ and $\gamma \in \{0.01, 0.1, 1, 10\}$.

Model selection uses 10 fold cross validation.

SVM results:
Best C = 1.0
Best γ = 1.0
Mean CV accuracy = 0.8450
Test accuracy = 0.8260
Test error probability P(error) = 0.1740
The learned decision boundary is circular and fits the ring structure well.

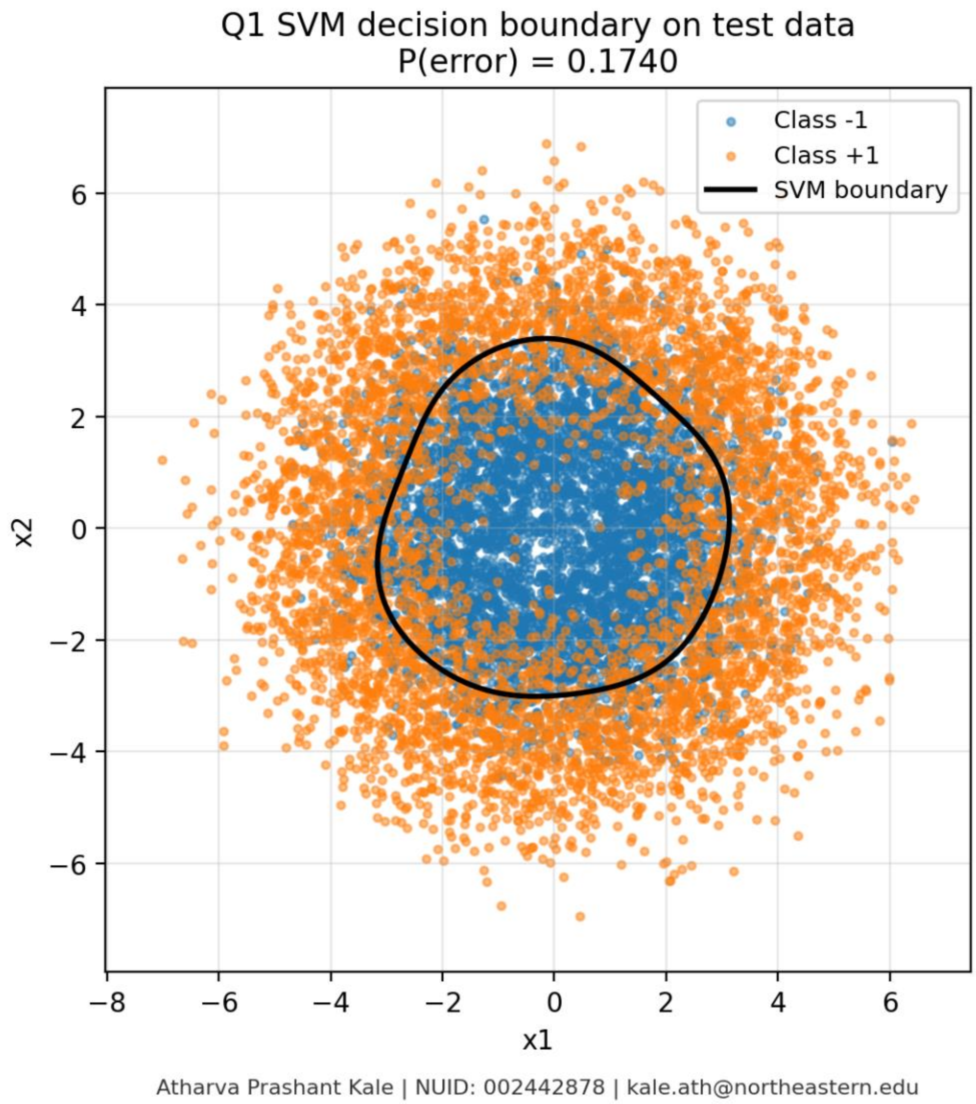

Figure 2: Q1 SVM decision boundary overlaid on the test samples from both classes.

2.3 Multilayer Perceptron Classifier

The MLP used has a single hidden layer with tanh activation.

Hidden layer computation:

$$\mathbf{h} = \mathbf{W}\,\mathbf{x} + \mathbf{b}$$

Activation:

$$\tanh(z) = (e^{z} - e^{-z}) / (e^{z} + e^{-z})$$

Output layer:

$$f(\mathbf{x}) = \mathbf{v}^{T} \tanh(\mathbf{h}) + \mathbf{c}$$

Hyperparameter search space:

$P \in \{4, 8, 16, 32, 64\}$

$\alpha \in \{0.0001, 0.001\}$

Training details:

Activation: tanh

Optimizer: Adam

Maximum iterations: 1000

Loss: Cross entropy

10 fold CV is performed for each (P, α) pair.

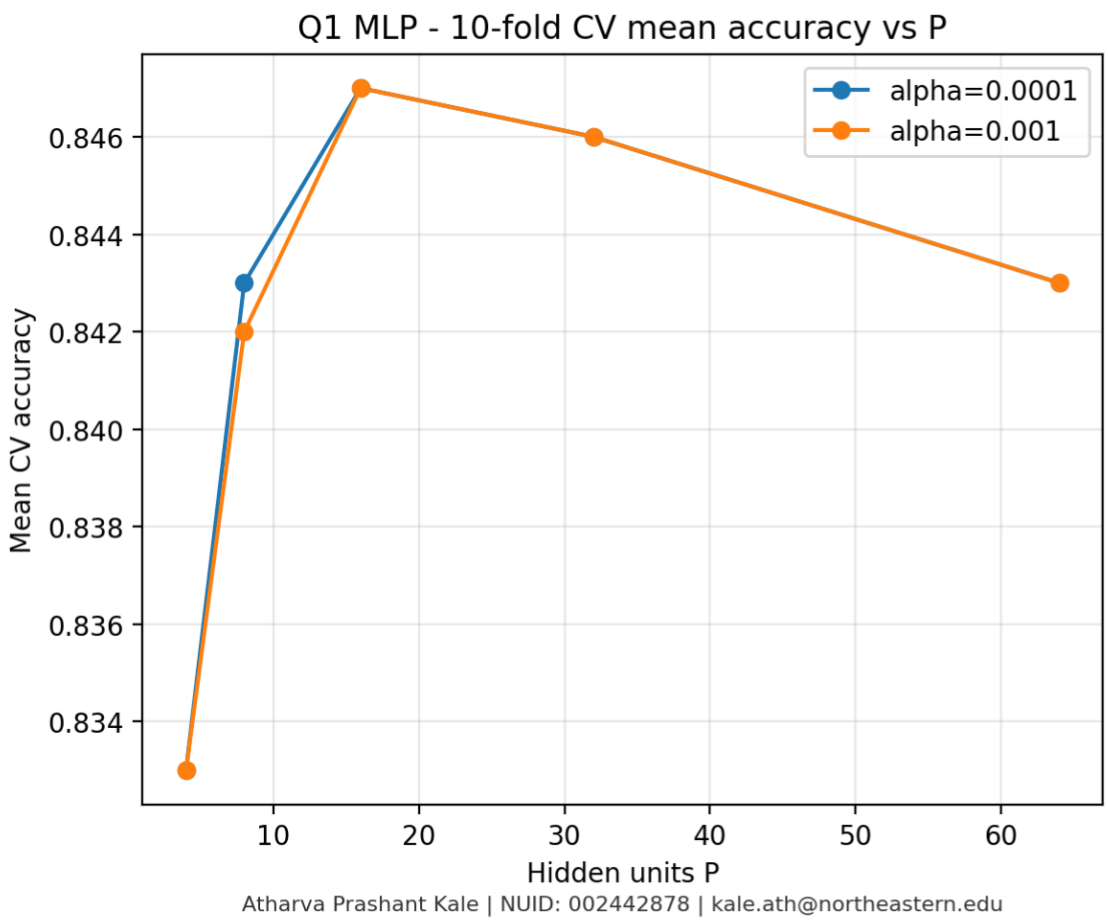

Figure 3: Q1 MLP ten fold cross validation accuracy as a function of hidden layer size P for α ∈ {1e−4, 1e−3}.

MLP results:
Best number of hidden units P = 16
Best regularization α = 0.0001
Mean CV accuracy = 0.8470
Test accuracy = 0.8243
Test error probability P(error) = 0.1757
The decision boundary learned by the MLP is smooth and circular, similar to the SVM.

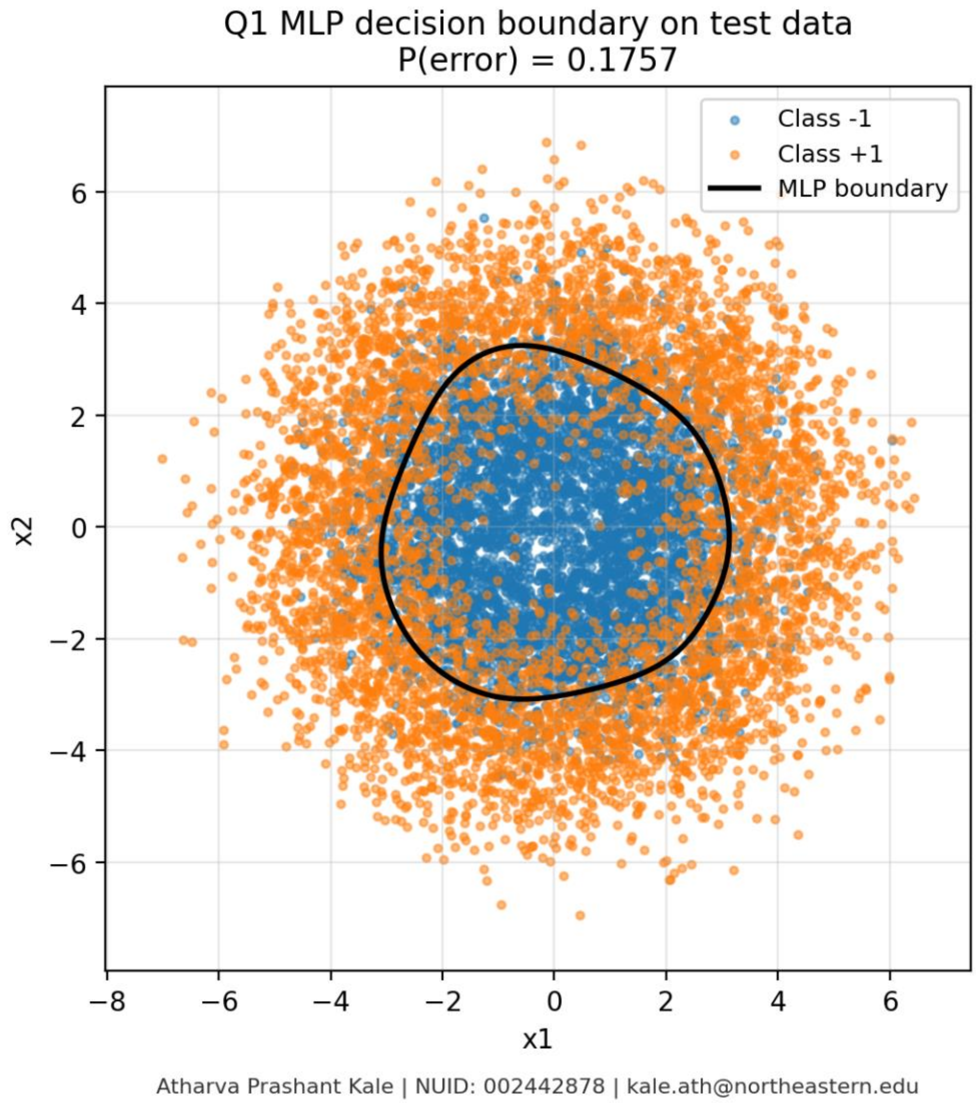

Figure 4: Q1 MLP decision boundary drawn as the contour where the predicted posterior probability for class +1 equals 0.5.

2.4 Comparison of SVM and MLP

SVM test error: 0.1740

MLP test error: 0.1757

Both achieve nearly identical performance because:

1. Both models can represent circular decision surfaces.
2. Both have enough capacity to fit the noisy data.
3. The noise level limits maximum accuracy.

Strengths of SVM:

• Convex optimization gives a unique global solution

• Very stable

• Excellent on small and medium sized datasets

Weakness:
• Kernel SVM scales poorly to very large datasets

Strengths of MLP:
• Highly flexible universal function approximator
• Scales to large datasets
• Can be extended to deeper architectures

Weaknesses:
• Training is non convex
• Requires tuning of learning rate and architecture
• Sensitive to initialization

Both methods successfully capture the nonlinear structure and achieve similar accuracy.

3. Question 2: Gaussian Mixture Model Based Image Segmentation

3.1 Objective

The goal is to segment a natural color image by clustering its pixels in a five dimensional feature space using a Gaussian Mixture Model. The number of mixture components K is selected using cross validation, and the final segmentation assigns each pixel to the component with maximum posterior probability.
This method performs unsupervised segmentation where the model discovers color and spatial patterns automatically.

3.2 Dataset and Preprocessing

The BSDS300 dataset contains 300 natural images with varied lighting, textures, edges, and color regions. We downloaded the archive, extracted it, and selected the first available image.

Image shape in this run:
Height H = 481
Width W = 321
Channels = 3 (RGB)

Figure 5: Q2 original BSDS300 test image used for Gaussian mixture segmentation.

Every pixel is converted into a five-dimensional feature vector:
f = [row, column, R, G, B]
This ensures that segmentation depends on both spatial location and color. Without spatial coordinates, the GMM might incorrectly cluster identical colors across far separated regions.

Each feature dimension is normalized independently to the range [0, 1]:

$$\hat{f} = (f - \min(f)) / (\max(f) - \min(f))$$

This prevents the row and column values from dominating the RGB values in the GMM likelihood computation.

For cross validation, at most 40000 randomly sampled pixels are used to reduce computation time. All pixels are used when fitting the final GMM.

3.3 Gaussian Mixture Model (GMM)

A GMM represents the density of data as a weighted sum of K Gaussian components:

$$p(x) = \Sigma_k \pi_k N(x \mid \mu_k, \Sigma_k)$$

where:

π_k is the mixing weight

μ_k is the mean of component k

Σ_k is the covariance matrix of component k

Each pixel feature $\hat{f}$ is modeled as a sample from this mixture distribution.

The parameters are estimated using maximum likelihood via the EM algorithm:

Expectation step:

Compute responsibilities

$$\gamma_k(i) = \pi_k N(x_i \mid \mu_k, \Sigma_k) / \Sigma_j \pi_j N(x_i \mid \mu_j, \Sigma_j)$$

Maximization step:

Update π_k, μ_k, Σ_k using the weighted sample averages.

The log likelihood for a validation set is:

$$L = (1 / N) \Sigma_i \log p(x_i)$$

The goal is to choose K that maximizes validation log likelihood.

3.4 Model Order Selection with Cross Validation

We test the following values of K:

$$K \in \{2, 3, 4, 5, 6\}$$

For each K:

1. Split the CV pixel set into 5 folds.
2. Train a GMM on 4 folds.
3. Compute mean log likelihood on the held out fold.
4. Average across all folds.

The K with the highest mean validation log likelihood is selected.

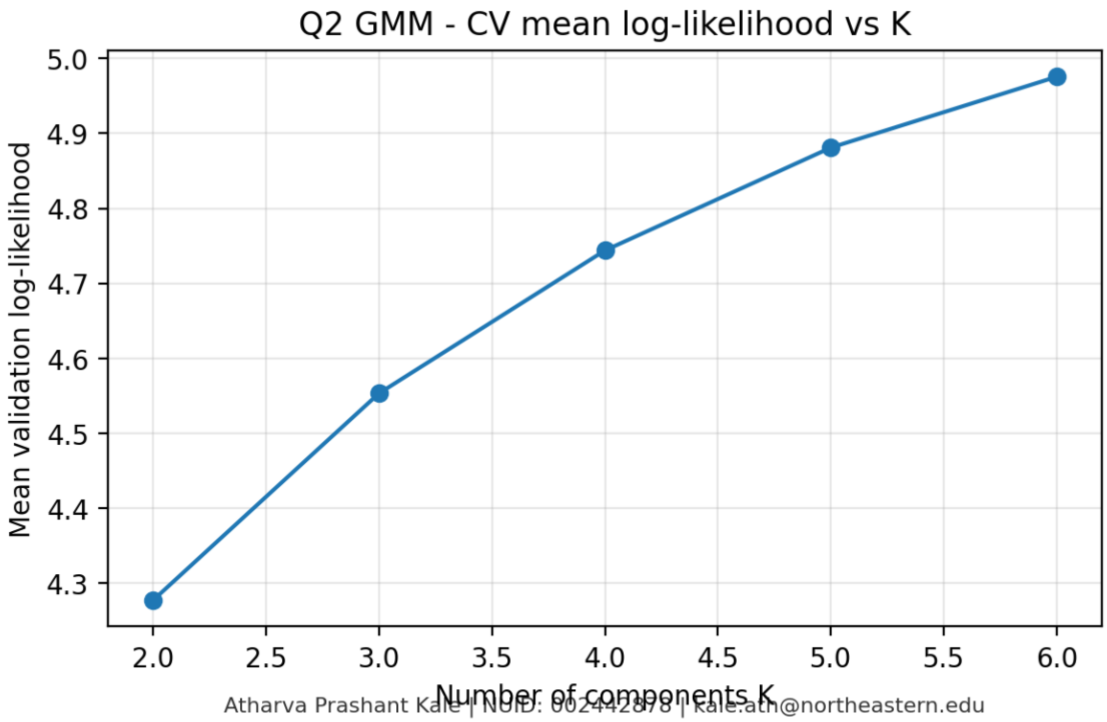

Figure 6: Q2 average validation log likelihood as a function of mixture order K, showing K = 6 as the maximizer.

Example results from this run:
K = 2 → mean log likelihood = 4.277
K = 3 → mean log likelihood = 4.553
K = 4 → mean log likelihood = 4.744
K = 5 → mean log likelihood = 4.881
K = 6 → mean log likelihood = 4.976

The performance keeps improving, so the best K is:

K* = 6

This indicates the image has approximately six natural clusters that best explain the pixel distribution.

3.5 Final Image Segmentation

A GMM with K = 6 is trained on all pixel features. For each pixel, we compute the posterior probability of belonging to each component:

Posterior:

$$\mathbf{p(k \mid x) = \pi_k N(x \mid \mu_k, \Sigma_k) / \Sigma_j \pi_j N(x \mid \mu_j, \Sigma_j)}$$

The assigned label is:

$$\mathbf{label(x) = argmax_k\ p(k \mid x)}$$

We then map each label to a grayscale intensity:

Component 1 → 0

Component 2 → 51

Component 3 → 102

Component 4 → 153

Component 5 → 204

Component 6 → 255

This produces a clean grayscale segmentation where regions belonging to the same GMM cluster appear with similar intensity.

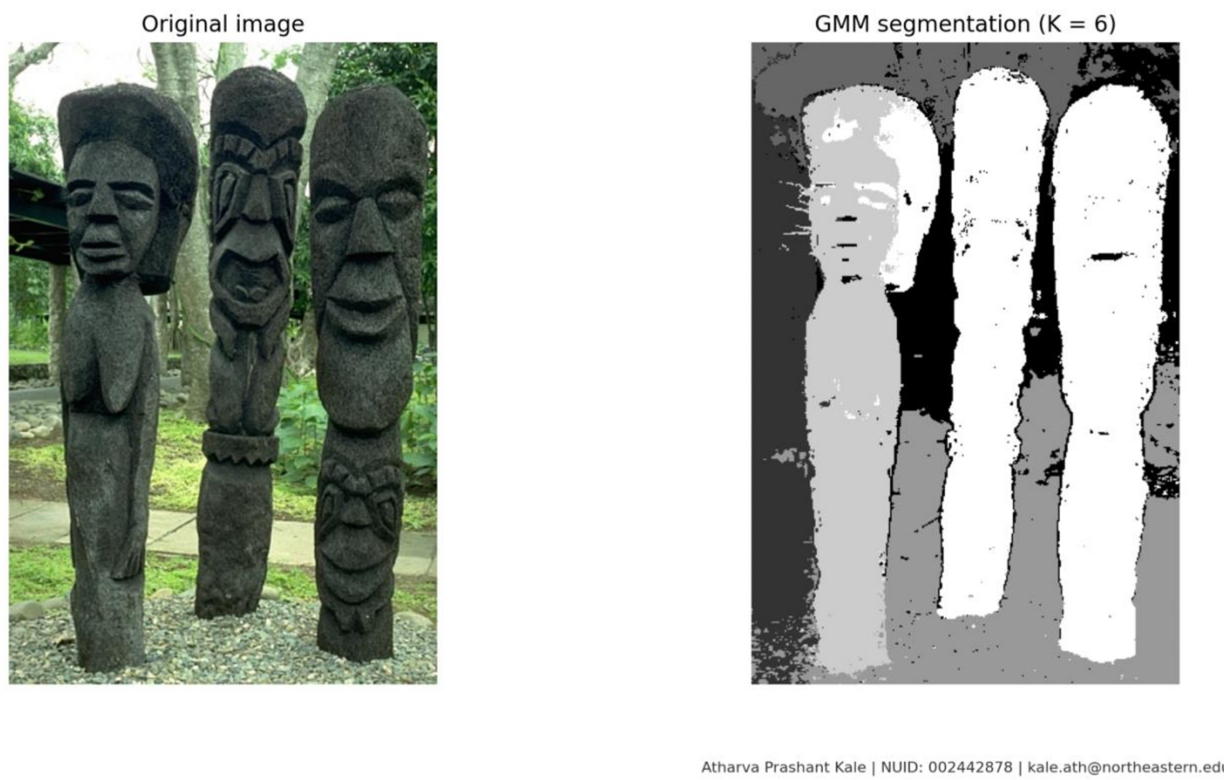

Figure 7: Q2 final Gaussian mixture segmentation result with K = 6 components, shown beside the original BSDS300 image.

3.6 Interpretation of Segmentation

The GMM separates regions based on color similarity and spatial coherence. For natural images, this tends to group:

- sky and background areas
- foliage and vegetation
- object boundaries
- shadows and highlights
- textured regions with similar chromaticity

The segmentation is unsupervised, so the model does not know the semantic meaning of objects. Instead, it identifies low level structures consistent with the image statistics.

The inclusion of spatial coordinates prevents disconnected but identically colored regions from merging. This gives smoother segmentations where clusters form spatially continuous regions.

3.7 Advantages and Limitations

Advantages:

- Fully unsupervised segmentation
- Flexible model capable of capturing complex multimodal color distributions
- Soft probabilistic clustering
- Uses both color and spatial information
- Model order selection avoids underfitting or overfitting

Limitations:

- EM can converge to local optima depending on initialization
- Complexity increases with K and with image size
- Segmentation does not enforce spatial smoothness beyond what the data naturally provides
- Results are sensitive to noise in uniform regions

3.8 Conclusion

The GMM based segmentation pipeline successfully identifies coherent regions in the BSDS300 image using a five dimensional feature representation. Cross validation selects $K = 6$, giving the best tradeoff between model complexity and generalization. The segmentation visually matches major structural regions of the image and demonstrates the effectiveness of probabilistic modeling for unsupervised image analysis.

4. Overall Assignment Conclusion

Overall, this assignment demonstrated how nonlinear supervised learning methods such as SVM with an RBF kernel and a single layer MLP can successfully model complex circular decision boundaries in synthetic data. Support Vector Machines offered stable performance with strong generalization, while the MLP achieved similar accuracy by learning the nonlinearity through its hidden layer. In the second task, a Gaussian Mixture Model with spatial and color based pixel features provided an effective unsupervised segmentation approach. Cross validation identified the optimal number of mixture components, and the resulting segmentation aligned well with visual structures in the BSDS300 image. Together, these experiments highlight the usefulness of probabilistic modeling and nonlinear classifiers in modern machine learning applications.

References

[1] C. M. Bishop, “Pattern Recognition and Machine Learning.” Springer, 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning.” Springer, 2nd ed., 2009.

[3] C. Cortes and V. Vapnik, “Support Vector Networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[4] S. Haykin, “Neural Networks and Learning Machines.” Prentice Hall, 3rd ed., 2008.

[5] G. McLachlan and D. Peel, “Finite Mixture Models.” John Wiley & Sons, 2000.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” Journal of the Royal Statistical Society: Series B, vol. 39, no. 1, pp. 1–38, 1977.

[7] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph based image segmentation,” International Journal of Computer Vision, vol. 59, no. 2, pp. 167–181, 2004.

[8] Berkeley Segmentation Dataset BSDS300, University of California, Berkeley, available at: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[9] Scikit-learn Machine Learning Library in Python, available at: https://scikit-learn.org/

[10] NumPy and Matplotlib Scientific Python Libraries, available at: https://numpy.org/ and **https://matplotlib.org/**

Code Availability

All code used for this assignment is available in the following public GitHub repository:

https://github.com/AtharvaK1810/EECE5644-Machine-Learning-and-Pattern-Recognition-AtharvaKale/tree/main/Assignment%204

The repository contains:

- The full Google Colab compatible Python script for Question 1 and Question 2.
- The generated output figures saved under a4_outputs/.
- A README file that explains how to run the script and reproduce the results.