HW4

Question 1 (50%)

Train and test Support Vector Machine (SVM) and Multi-layer Perceptron (MLP) classifiers that aim for minimum probability of classification error (i.e. we are using 0-1 loss; all error instances are equally bad). You may use a trusted implementation of training, validation, and testing in your choice of programming language. The SVM should use a Gaussian (sometimes called radial-basis) kernel. The MLP should be a single-hidden layer model with your choice of activation functions for all perceptrons.

Generate 1000 independent and identically distributed (iid) samples for training and 10000 iid samples for testing. All data for class $l \in \{-1, +1\}$ should be generated as follows:

$$\mathbf{x} = r_l \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} + \mathbf{n} \tag{1}$$

where $\theta \sim Uniform[-\pi, \pi]$ and $\mathbf{n} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$. Use $r_{-1} = 2, r_{+1} = 4, \sigma = 1$.

Note: The two class sample sets will be highly overlapping two concentric disks, and due to angular symmetry, we anticipate the best classification boundary to be a circle between the two disks. Your SVM and MLP models will try to approximate it. Since the optimal boundary is expected to be a quadratic curve, quadratic polynomial activation functions in the hidden layer of the MLP may be considered as to be an appropriate modeling choice. If you have time (optional, not needed for assignment), experiment with different activation function selections to see the effect of this choice.

Use the training data with 10-fold cross-validation to determine the best hyperparameters (box constraints parameter and Gaussian kernel width for the SVM, number of perceptrons in the hidden layer for the MLP). Once these hyperparameters are set, train your final SVM and MLP classifier using the entire training data set. Apply your trained SVM and MLP classifiers to the test data set and estimate the probability of error from this data set.

Report the following: (1) visual and numerical demonstrations of the K-fold cross-validation process indicating how the hyperparameters for SVM and MLP classifiers are set; (2) visual and numerical demonstrations of the performance of your SVM and MLP classifiers on the test data set. It is your responsibility to figure out how to present your results in a convincing fashion to indicate the quality of training procedure execution, and the test performance estimate.

Hint: For hyperparameter selection, you may show the performance estimates for various choices and indicate where the best result is achieved. For test performance, you may show the data and classification boundary superimposed, along with an estimated probability of error from the samples. Modify and supplement these ideas as you see appropriate.

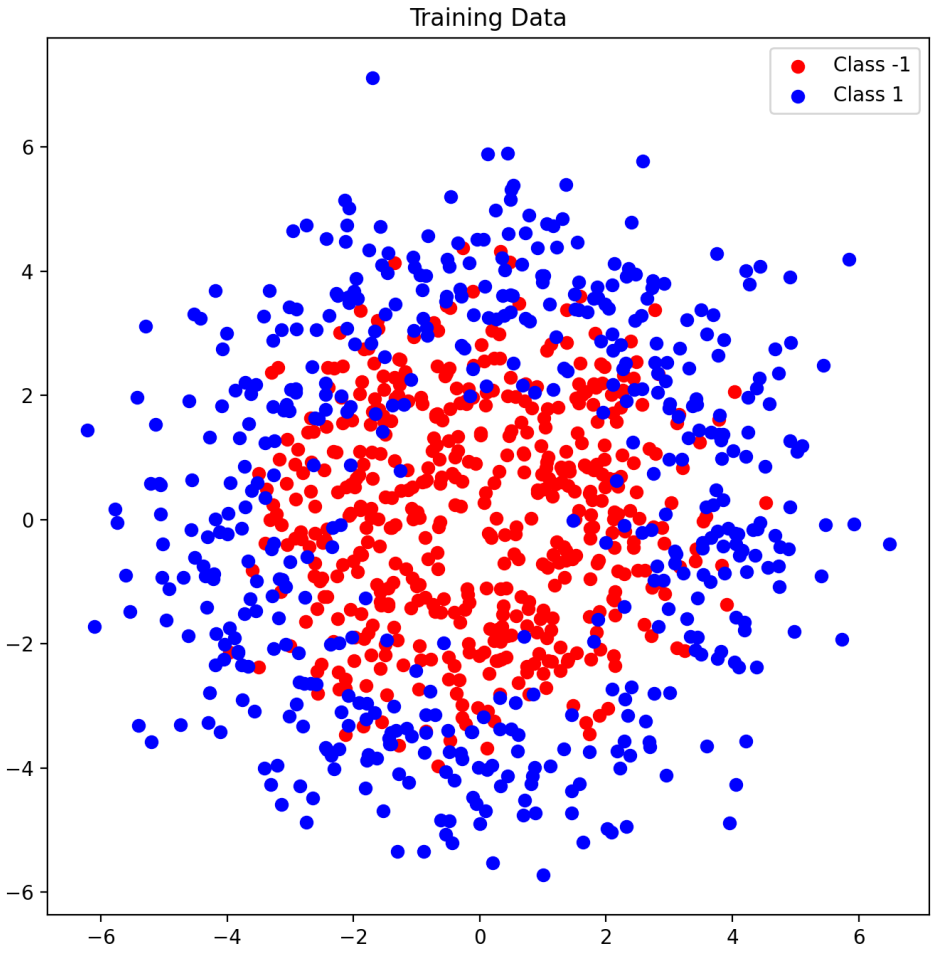
Generated two concentric data classes with one using radius of 2 and the other with radius of 4.

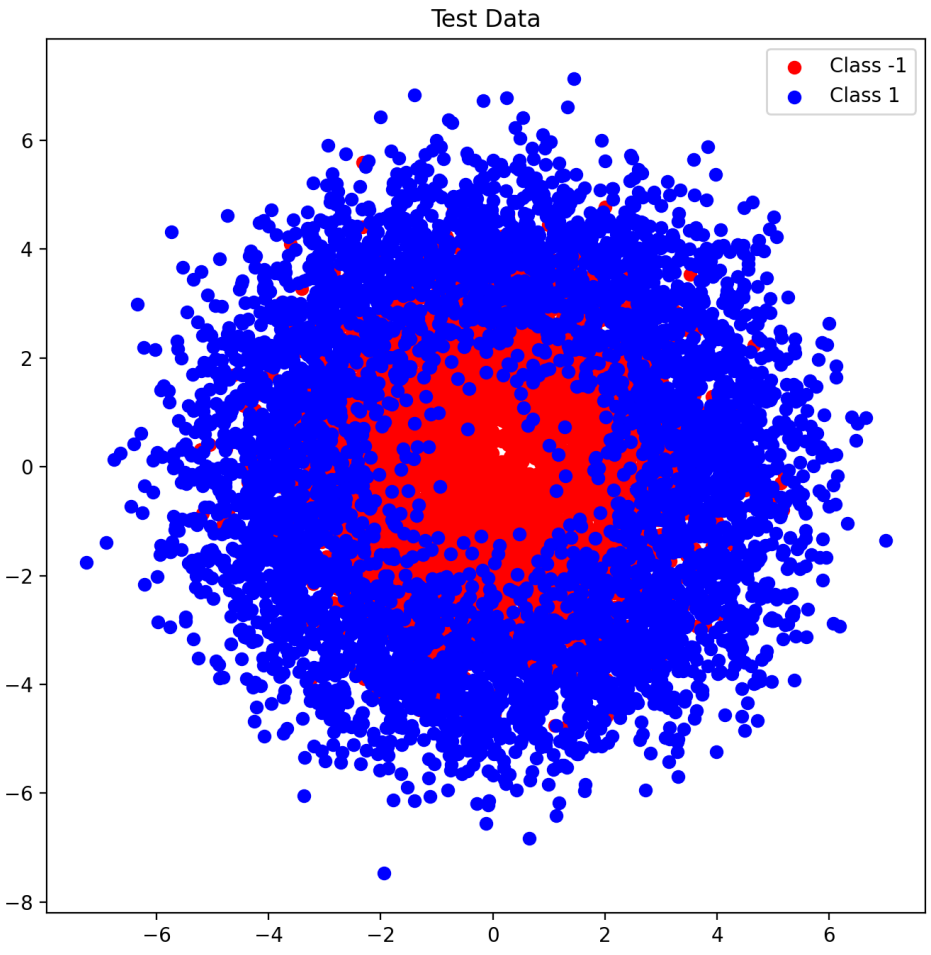
Classified data using both an SVM and MLP, results are below.

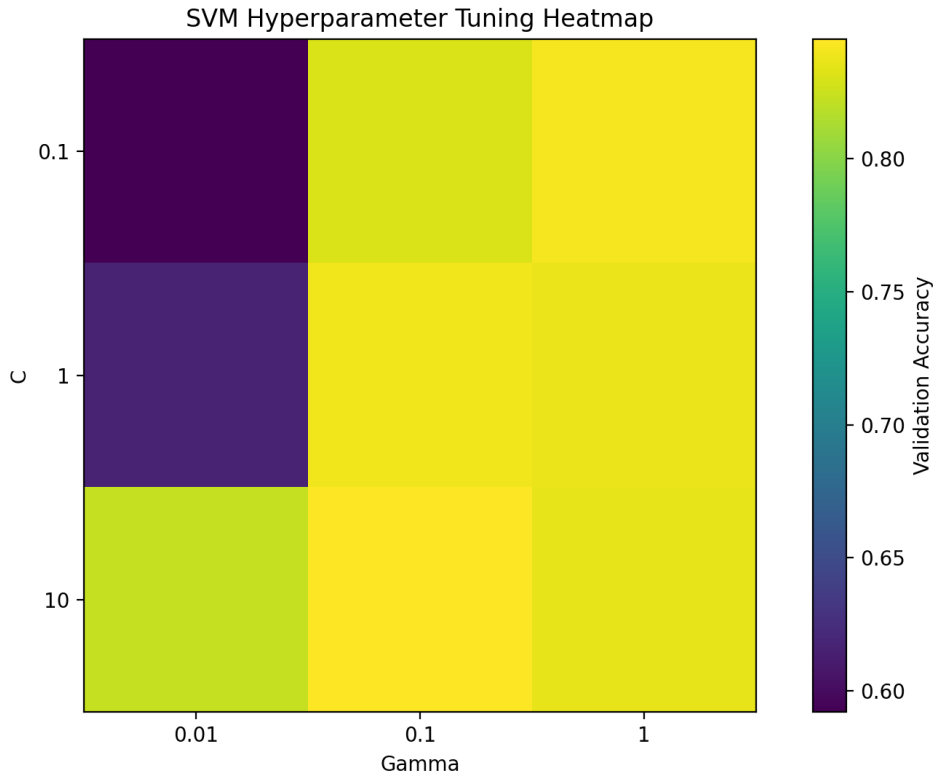
The best hyperparameters were chose for the SVM using the GridSearchCV function which returns a “best_estimator” that can then be used for the SVM.

MLP:

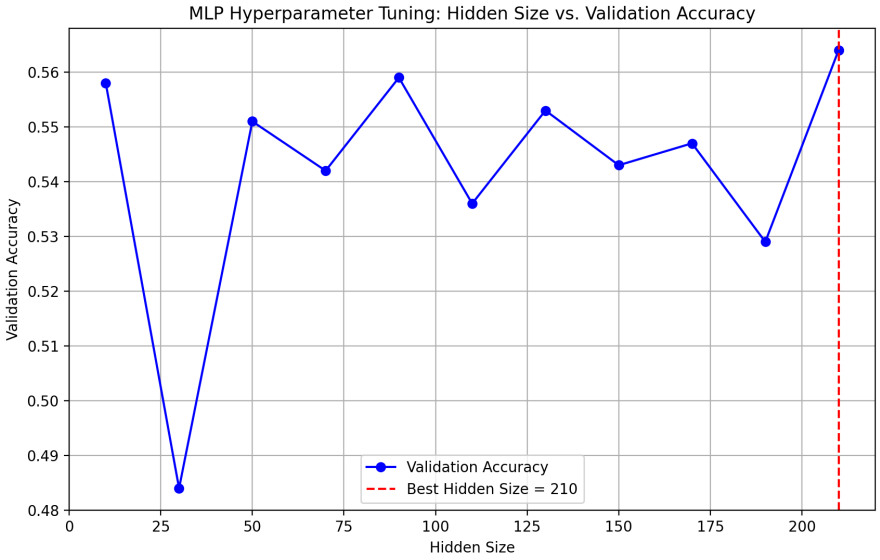
Using quadratic activation function since expected boundary is a circle.


Training Data
Class -1
Class 1
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


Test Data
Class -1
Class 1
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6



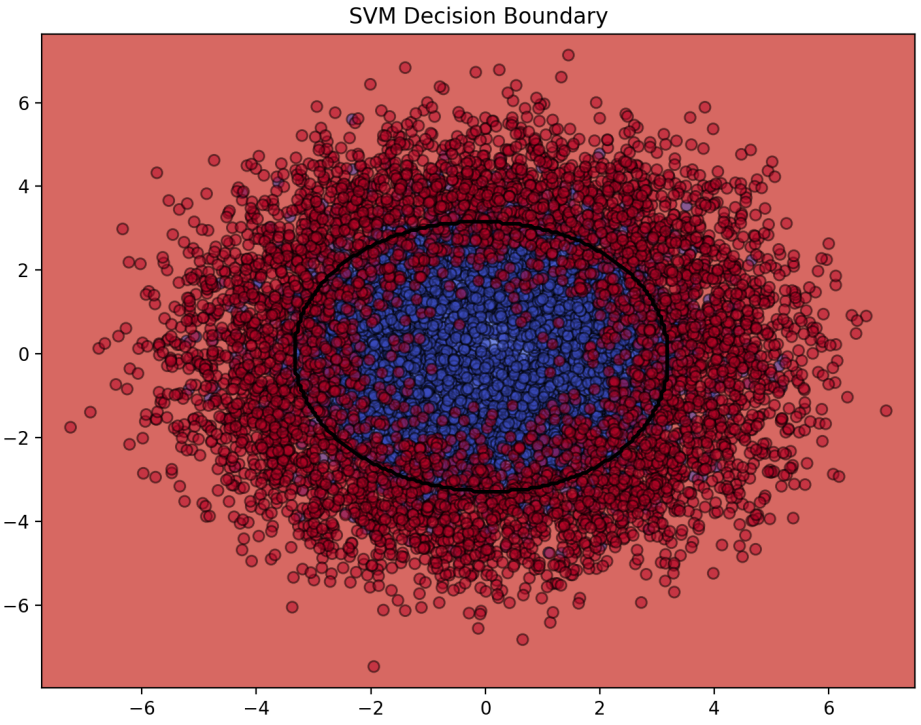
Best SVM Parameters: {'C': 10, 'gamma': 0.1}
SVM Test Accuracy: 82.76%

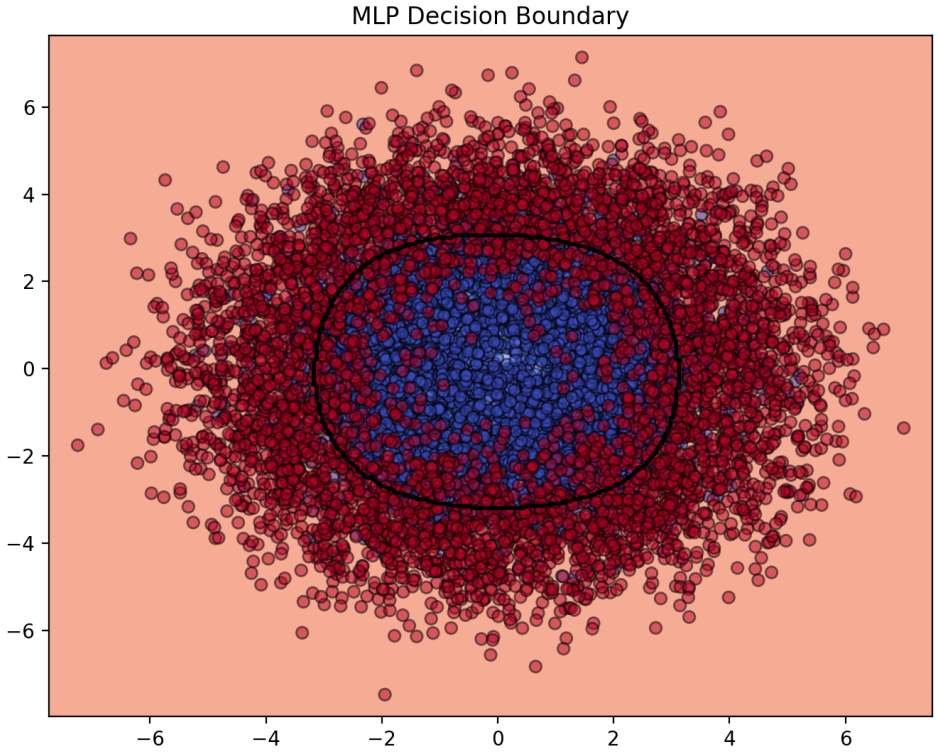


Different hidden layer sizes were tried and 210 perceptrons had the best results, so this was used for training.

Hidden Size: 10, Average Validation Accuracy: 55.80%
Hidden Size: 30, Average Validation Accuracy: 48.40%
Hidden Size: 50, Average Validation Accuracy: 55.10%

Hidden Size: 70, Average Validation Accuracy: 54.20%
Hidden Size: 90, Average Validation Accuracy: 55.90%
Hidden Size: 110, Average Validation Accuracy: 53.60%
Hidden Size: 130, Average Validation Accuracy: 55.30%
Hidden Size: 150, Average Validation Accuracy: 54.30%
Hidden Size: 170, Average Validation Accuracy: 54.70%
Hidden Size: 190, Average Validation Accuracy: 52.90%
Hidden Size: 210, Average Validation Accuracy: 56.40%
Best Hidden Size: 210, Best Validation Accuracy: 56.40%





Epoch 0, Loss: 0.6813945770263672, Train Accuracy: 52.90%
Epoch 100, Loss: 0.6419117450714111, Train Accuracy: 50.00%
Epoch 200, Loss: 0.6144750714302063, Train Accuracy: 61.40%
Epoch 300, Loss: 0.5889546871185303, Train Accuracy: 70.40%
Epoch 400, Loss: 0.5642116665840149, Train Accuracy: 76.10%
Epoch 500, Loss: 0.5401427149772644, Train Accuracy: 79.00%
Epoch 600, Loss: 0.5169431567192078, Train Accuracy: 81.00%
Epoch 700, Loss: 0.49491870403289795, Train Accuracy: 82.10%
Epoch 800, Loss: 0.4743998944759369, Train Accuracy: 82.70%
Epoch 900, Loss: 0.4556710720062256, Train Accuracy: 82.90%
Epoch 1000, Loss: 0.4389151632785797, Train Accuracy: 83.30%
Epoch 1100, Loss: 0.424197256565094, Train Accuracy: 83.50%
Epoch 1200, Loss: 0.411477267742157, Train Accuracy: 83.90%
Epoch 1300, Loss: 0.400636225938797, Train Accuracy: 84.00%
Epoch 1400, Loss: 0.39150547981262207, Train Accuracy: 84.30%
Epoch 1500, Loss: 0.38389328122138977, Train Accuracy: 84.70%
Epoch 1600, Loss: 0.3776039481163025, Train Accuracy: 84.80%
Epoch 1700, Loss: 0.372449666261673, Train Accuracy: 84.90%
Epoch 1800, Loss: 0.36825793981552124, Train Accuracy: 85.10%
Epoch 1900, Loss: 0.3648737668991089, Train Accuracy: 85.10%
MLP Test Accuracy: 82.83%

Question 2 (50%)

In this question, you will use GMM-based clustering to segment a color image. Pick your color image from this dataset: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/html/dataset/images.html.

1

As preprocessing, for each pixel, generate a 5-dimensional feature vector as follows: (1) append row index, column index, red value, green value, blue value for each pixel into a raw feature vector; (2) normalize each feature entry individually to the interval $[0, 1]$, so that all of the feature vectors representing every pixel in an image fit into the 5-dimensional unit-hypercube.

Fit a Gaussian Mixture Model to these normalized feature vectors representing the pixels of the image. To fit the GMM, use maximum likelihood parameter estimation and 10-fold cross-validation (with maximum average validation-log-likelihood as the objective) for model order selection.

Once you have identified the *best* GMM for your feature ectors, assign the most likely component label to each pixel by evaluating component label posterior probabilities for each feature vector according to your GMM, similar to MAP classification. Present the original image and your GMM-based segmentation labels assigned to each pixel side by side for easy visual assessment of your segmentation outcome. If using grayscale values as segment/component labels, please uniformly distribute them between min/max grayscale values to have good contrast in the label image.

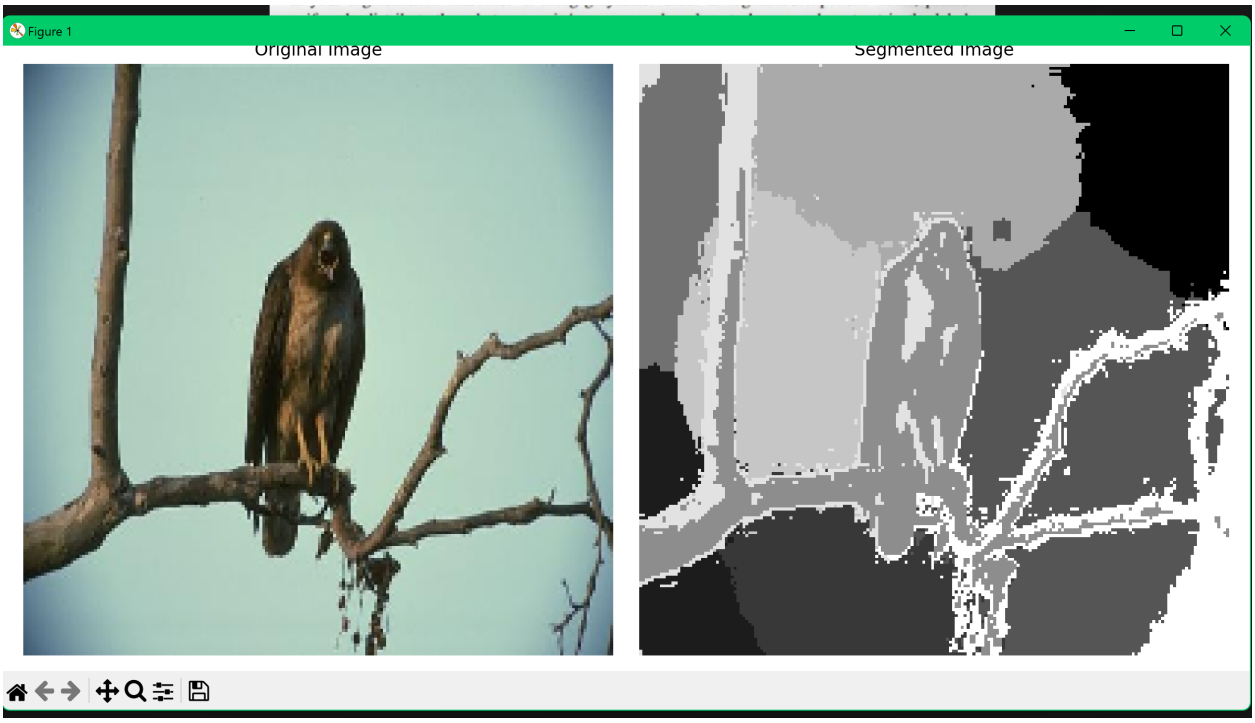
Hint: If the image has too many pixels for your available computational power, you may downsample the image to reduce overall computational needs).

https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/BSDS300/html/dataset/images.html

First a color image was selected:

Best number of components: 10

Output image:



- **Feature Extraction**: Each pixel was represented as a 5-dimensional feature vector: row index, column index, and normalized RGB values.
- **Normalization**: All feature dimensions were scaled to the range [0,1] for consistency.

  [0,1][0, 1]
- **GMM Training**: A GMM was fit using maximum likelihood estimation, with cross-validation (10-fold) applied for model order selection.
- **Pixel Labeling**: Each pixel was assigned the most likely GMM component label based on posterior probabilities (MAP classification).
- **Visualization**: Segmentation labels were mapped to grayscale values for contrast, and the original and segmented images were presented side by side.