Question 1 (60%)

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 in your choice of programming language and software packages, e.g., Python coupled with the PyTorch library for MLPs or scikit-learn for SVMs. 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 function 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 $\theta \sim \text{Uniform}[-\pi, \pi]$ and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$. Use $r_{-1} = 2, r_{+1} = 4, \sigma = 1$.

Note: The two class sample sets will be two highly overlapping 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 an appropriate modeling choice. If you have time (optional; no bonus marks), experiment with different activation functions to see the effect of this choice.

Use 10-fold cross-validation on the training data to determine the best hyperparameters. For SVMs, these will be the box constraints or regularization parameter on the max-margin objective to prevent overfitting, λ as in lecture notes or sometimes denoted as C, as well as the Gaussian kernel bandwidth, $\gamma = \frac{1}{2\sigma^2}$. For the MLP, this will be the number of perceptrons in the hidden layer. 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.

*Note: When performing hyperparameter selection on combinations of hyperparameters, as in the SVM where you wish to select the best C **and** γ, then it is common to perform a grid-search over all possible combinations of hyperparameters. Whilst this is computationally expensive, as you are exploring all possible combinations of the hyperparameters, it ensures that you have the best model setting. Please look at the GridSearchCV class and the user guide for more information.*

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 the training/model selection procedure 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.

Question 2 (40%)

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`.

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 (GMM) to these normalized feature vectors representing the pixels of the image. To fit the GMM, use maximum likelihood parameter estimation (e.g., via Expectation Maximisation as in the ”fit” function of the GaussianMixture class) and 10-fold cross-validation with maximum average validation-log-likelihood as the objective for model order selection[1]. Once you have identified the *best* GMM for your feature vectors, assign the most likely component label (MAP) to each pixel by evaluating component label posterior probabilities for each feature vector. 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).

[1]For K-means, using cross-validation with distortion as the validation objective would be a poor method of choosing K, as each cluster is represented as a “degenerative spike”, which is only more likely to accurately capture the data as K increases.