EECE5644 Fall 2020 – Take Home Exam 2

Submit: Friday, 2020-October-30 before 10:00am ET

Please submit your solutions on Canvas in a single PDF file that includes all math, numerical and visual results, and code (as appendix or as a link to an online code repository). Only the contents of this PDF will be graded. Do not link from the pdf to external documents online where results may be presented, as all contents outside the PDF will be ignored for grading purposes.

This is a graded assignment and the entirety of your submission must contain only your own work. You may benefit from publicly available literature including software (not from classmates), as long as these sources are properly acknowledged in your submission.

Copying math or code from each other are not allowed and will be considered as academic dishonesty, and will be treated accordingly. There cannot be any written material exchange between classmates, but verbal discussions with others to get help or ideas are acceptable. Discussing with the instructor, the teaching assistant, and classmates at open office periods to get clarification or to eliminate doubts are acceptable. Anything discussed and recorded in class or office periods can be used when constructing your own solution.

By submitting a PDF file in response to this take home assignment you are declaring that the contents of your submission, and the associated code is your own work, except as noted in your citations to resources.

Question 1 (50%)

The probability density function (pdf) for a 2-dimensional real-valued random vector $\mathbf{X}$ is as follows: $p(\mathbf{x}) = p(\mathbf{x}|L=0)P(L=0) + p(\mathbf{x}|L=1)P(L=1)$. Here L is the true class label that indicates which class-label-conditioned pdf generates the data.

The class priors are $P(L=0) = 0.6$ and $P(L=1) = 0.4$. The class class-conditional pdfs are $p(\mathbf{x}|L=0) = w_1 g(\mathbf{x}|\mathbf{m}_{01}, \mathbf{C}_{01}) + w_2 g(\mathbf{x}|\mathbf{m}_{02}, \mathbf{C}_{02})$ and $p(\mathbf{x}|L=1) = g(\mathbf{x}|\mathbf{m}_1, \mathbf{C}_1)$, where $g(\mathbf{x}|\mathbf{m}, \mathbf{C})$ is a multivariate Gaussian probability density function with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{C}$. The parameters of the class-conditional Gaussian pdfs are: $w_1 = w_2 = 1/2$, and

$$\mathbf{m}_{01} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad \mathbf{C}_{01} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \quad \mathbf{m}_{02} = \begin{bmatrix} 0 \\ 4 \end{bmatrix} \quad \mathbf{C}_{02} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad \mathbf{m}_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

For numerical results requested below, generate the following independent datasets each consisting of iid samples from the specified data distribution, and in each dataset make sure to include the true class label for each sample. Save the data and use the same data set in all subsequent exercises.

- D_{train}^{100} consists of 100 samples and their labels for training;
- D_{train}^{1000} consists of 1000 samples and their labels for training;
- D_{train}^{10000} consists of 10000 samples and their labels for training;
- $D_{validate}^{20K}$ consists of 20000 samples and their labels for validation;

Part 1: (10%) Determine the theoretically optimal classifier that achieves minimum probability of error using the knowledge of the true pdf. Specify the classifier mathematically and implement it; then apply it to all samples in $D_{validate}^{20K}$. From the decision results and true labels for this validation set, estimate and plot the ROC curve of this min-P(error) classifier, and on the ROC curve indicate, with a special marker, indicate the point that corresponds to the min-P(error) classifier's operating point. Also report your estimate of the min-P(error) achievable, based on counts of decision-truth label pairs on $D_{validate}^{20K}$. Optional: As supplementary visualization, generate a plot of the decision boundary of this classification rule overlaid on the validation dataset. This establishes an aspirational performance level on this data for the following approximations.

Part 2: (20%) (a) Using the maximum likelihood parameter estimation technique, estimate the class priors and class conditional pdfs using training data in D_{train}^{10000}. As class conditional pdf models, for $L=0$ use a Gaussian Mixture model with 2 components, and for $L=1$ use a single Gaussian pdf model. For each estimated parameter, specify the maximum-likelihood-estimation objective function that is maximized as well as the iterative numerical optimization procedure used, or if applicable, for analytically tractable parameter estimates, specify the estimator formula. Using these estimated class priors and pdfs, design and implement an approximation of the min-P(error) classifier, apply it on the validation dataset $D_{validate}^{20K}$. Report the ROC curve and minimum probability of error achieved on the validation dataset with this classifier that is trained with 10000 samples. (b) Repeat Part (2a) using D_{train}^{1000} as the training dataset. (c) Repeat Part (2a) using D_{train}^{100} as the training dataset. How does the performance of your approximate min-P(error) classifier change as the model parameters are estimated (trained) using fewer samples?

Part 3: (20%) (a) Using the maximum likelihood parameter estimation technique train a logistic-linear-function-based approximation of class label posterior function given a sample. As in part 2, repeat the training process for each of the three training sets to see the effect of training set sample count; use the validation set for performance assessment in each case. When optimizing the

parameters, specify the optimization problem as minimization of the negative-log-likelihood of the training dataset, and use your favorite numerical optimization approach, such as gradient descent or Matlab's *fminsearch* or Python's *minimize*. Use the trained class-label-posterior approximations to classify validation samples to approximate the minimum-P(error) classification rule; estimate the probability of error that these three classifiers attain using counts of decisions on the validation set. Optional: As supplementary visualization, generate plots of the decision boundaries of these trained classifiers superimposed on their respective training datasets and the validation dataset. (b) Repeat the process described in Part (3a) using a logistic-quadratic-function-based approximation of class label posterior functions given a sample.

Note 1: With $\mathbf{x}$ representing the input sample vector and $\mathbf{w}$ denoting the model parameter vector, logistic-linear-function refers to $h(\mathbf{x},\mathbf{w}) = 1/(1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x})})$, where $\mathbf{z}(\mathbf{x}) = [1,\mathbf{x}^{\mathrm{T}}]^{\mathrm{T}}$; and logistic-quadratic-function refers to $h(\mathbf{x},\mathbf{w}) = 1/(1+e^{-\mathbf{w}^{\mathrm{T}}\mathbf{z}(\mathbf{x})})$, where $\mathbf{z}(\mathbf{x}) = [1,x_1,x_2,x_1^2,x_1x_2,x_2^2]^{\mathrm{T}}$.

Hint: The classifier designed in Part 1 uses the true pdf knowledge and achieves theoretically optimum performance in terms of minimizing P(error). The classifiers designed in Part 2 are approximations of the theoretically optimum classification rule using the correct functional form of the data pdf, however the quality of approximation for these generative models will get worse as their parameters are estimated with fewer training samples. The classifiers in Part 3 attempt to directly approximate class label posteriors, and the approximtion capability increases as the model gets more complex (linear to quadratic). The classifiers in Part 3, however, are limited by the approximation capability of their functional form. While the classifiers in Part 2 can asymptotically approach the performance level of the theoretically optimal one if they are trained with more data, the classifiers in Part 3 are bounded in performance by the fact that they can only generate linear or quadratic decision boundaries, so no amount of training data will enable them to asymptotically approximate the theoretically optimal classification rule (which has a classification boundary that is more complex than quadratic).

Question 2 (50%)

The probability density function (pdf) for a 2-dimensional real-valued random vector $\mathbf{X}$ is a mixture of C Gaussian components with equal weights, distinct circularly symmetric covariance matrices, and mean vectors spaced evenly on a line. You may choose your own $C >= 10$, mean vectors, and covariance matrices for this problem.

Conduct the following model order selection exercise multiple (E) times (e.g., $E >= 100$), using both BIC and K-fold cross-validation (e.g., K = 10). Report a comprehensive (but compactly presented) summary of your experimental results regarding how BIC and K-fold cross-validation based model order selection decides on estimates of C (e.g., for both methods, show the histograms of estimated C values over multiple experiments).

Repeat the following E times (i.e. conduct multiple experiments) using different values of M, number of components in your Gaussian Mixture Model (GMM) of the data pdf (e.g. M from 1 to 20 or higher if needed),...

Generate multiple datasets from the true Gaussian mixture pdf you selected. Select your dataset sample counts as powers of 10 (e.g. from 10^2 to 10^6). Then, for each training set:

BIC (25%): For each M and for each dataset, use the EM algorithm to estimate the maximum likelihood parameter values for the M-component GMM, then evaluate the BIC for each M us-

ing the respective MLE-based GMMs. Select the best M (with smallest BIC value). Record the selected M values for each dataset in this experiment.

K-fold cross-validation (25%): For each dataset, using your selected K, partition the dataset appropriately, then leaving each partition out once as validation set, do the following for each M: (a) use the EM algorithm to estimate the maximum likelihood parameter values for the for the M-component GMM with the training partitions, (b) then evaluate the log-likelihood of the data in the validation partition using the trained M-component GMM, (c) compute the weighted average of validation-log-likelihood values across K partitions (weighted proportional to sample counts in each validation partition, so you end up with a sample-average of log-likelihood where each sample contributes equally to the validation performance measure). Now that you have the average log-likelihood value for M-component GMMs with K-fold cross-validation procedure, select the best M as the one that maximizes this cross-validation log-likelihood value. Record the selected M values for each dataset in this experiment.

After you run multiple experiments in this fashion, each with multiple datasets with different sample counts, count the number of times each M got selected using BIC and using K-fold cross-validation procedures, as the best model order estimate for each dataset. Summarize your results demonstrating the effect of dataset size on your model order selection for both of these model selection approaches. For instance you may report curves such as minimum/median/maximum M selected across experiments versus N, dataset sample count. These will provide statistical range/median information for model order selected using both methods depending on the number of samples used. You can make the plot richer by including multiple percentile curves (e.g., 10, 25, 50, 75, 90 percentile curves). Discuss how number of available samples in the dataset impact model order selection for both BIC and K-fold cross-validation methods.

Hint: You can use existing EM methods for GMMs in Matlab (fitgmdist) or Python. Implement your own BIC-objective function evaluation code and your own K-fold cross-validation average log-likelihood calculation code. Note that this code will have a lot of nested for loops for experiments, training set sample counts, model order, and for K-fold cross-validation, data partitions. Running the entire code may take a long time so use parallelization if you can. If you cannot parallelize, write the code and run for a few experiments (e.g. $E = 10$) and keep adding results in increments of small experiment counts, saving results cumulatively along the way. Do not discard results and start over, if the results can contribute to your final experiment summary appropriately.