EECE5644 Summer1 2021 – Assignment 3

Submit: Monday, 2021-June-14 before 10:00am ET

Please submit your solutions on Canvas in a single PDF file that includes all math, numerical and visual results. Either include a link to your code in an online repository or include the code as an appendix in the PDF file. The code is not graded, but helps verify your results are feasible as claimed. Only results and discussion presented in the PDF will be graded, so do not link to an external location where further results may be presented.

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. All discussions and materials shared during office periods are also acceptable resources and these tend to be very useful, so participate in office periods or take a look at their recordings. Cite your sources as appropriate.

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%)

In this exercise, you will train many multilayer perceptrons (MLP) to approximate the class label posteriors, using maximum likelihood parameter estimation (equivalently, with minimum average cross-entropy loss) to train the MLP. Then, you will use the trained models to approximate a MAP classification rule in an attempt to achieve minimum probability of error (i.e. to minimize expected loss with 0-1 loss assignments to correct-incorrect decisions).

Data Distribution: The data comes from $C = 4$ classes with uniform priors, and Gaussian class-conditional pdfs, as specified next for a 3-dimensional real-valued random vector $\mathbf{x}$. Consider the cube centered at the origin, faces parallel to the planes spanned by pairs of canonical bases, and an edge length of 2. The eight vertices of this cube are located at $[\pm 1, \pm 1, \pm 1]^T$. Each class conditional data pdf is a mixture of two Gaussian componenets. For each class conditional, designate two of the vertices of the cube as mean vectors of components. Assume equal (0.5) probability for each component. For each Gaussian component pick a covariance matrix in the form $\lambda \mathbf{I}$ where each component's λ value is different. Suggested values for these parameters are in the vicinity of 0.5 to 0.8 (to have some significant overla between GMM class conditional pdfs of each class).

MLP Structure: Use a 2-layer MLP (one hidden layer of perceptrons) that has P perceptrons in the first (hidden) layer with smooth-ramp style activation functions (e.g., Smooth-ReLU, ELU, etc, if your software package allows it; otherwise a sigmoid like the logistic function is also fine). At the second/output layer use a softmax function to ensure all outputs are positive and add up to 1. The best number of perceptrons for your custom problem will be selected using K-fold cross-validation.

Generate Data: Using your specified data distribution, generate multiple datasets: Training datasets with $100, 200, 500, 1000, 2000, 5000$ samples and a test dataset with 100000 samples. You will use the test dataset only for final performance evaluation of each trained model.

Theoretically Optimal Classifier: Using the knowledge of your true data pdf, construct the minimum-probability-of-error classification rule, apply it on the test dataset, and empirically estimate the probability of error for this theoretically optimal classifier. This provides the aspirational performance level for the MLP classfier.

Model Order Selection: For each of the training sets with different number of samples, perform 5-fold cross-validation, using minimum classification error probability as the objective function, to select the best number of perceptrons (that is justified by available training data). Make sure to NOT contaminate your model selection and training process with the test data; only use the provided training data samples in this cross-validation process. Since there are 6 different training datasets, you should end up with 6 separate P values obtained using cross-validation on their respective training sets.

Model Training: For each training set, having identified the best number of perceptrons using cross-validation, using maximum likelihood parameter estimation (minimum cross-entropy loss) train an MLP with the appropriate number of perceptrons using the entire respective training set. These are your final trained MLP models for class posteriors (possibly each with different number of perceptrons and certainly with different weights). In all of these training processes, make sure to mitigate the chances of getting stuck at a local optimum by randomly reinitializing each MLP training routine multiple times and getting the best solution you encounter among these.

Performance Assessment: Using each trained MLP as a model for class posteriors, and using the MAP decision rule (aiming to minimize the probability of error) classify the samples in the test set and for each trained MLP empirically estimate the probability of error.

Report Process and Results: Describe your process of developing the solution; numerically and visually report the test set empirical probability of error estimates for the theoretically optimal and multiple trained MLP classifiers. For instance show a plot of the empirically estimated test P(error) for each trained MLP versus number of training samples used in optimizing it (with semilog-x axis), as well as a horizontal line that runs across the plot indicating the empirically estimated test P(error) for the theoretically optimal classifier.

Note: You may use software packages for all aspects of your implementation. Make sure you use tools correctly. Explain in your report how you ensured the software tools do exactly what you need them to do.

Question 2 (50%)

The probability density function (pdf) for a 2-dimensional real-valued random vector **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 $C >= 5$, 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 with different data set realizations) using different values of M, number of components in your Gaussian Mixture Model (GMM) of the data pdf (e.g. M from 1 to 10 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 using 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 Python (sklearn.mixture.GaussianMixture) or Matlab (fitgmdist). 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, keep saving results and adding on to them with more experiments.