EECE5644 Spring 2021 – Assignment 3

Submit: Monday, 2021-April-05 before 10:00am ET

There will not be any extensions! Assignment 4 might become available before this deadline, so try to finish early.

Please submit your solutions on Canvas in a single PDF file that includes all math, numerical/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. Any content outside the PDF will be ignored for grading purposes. External code repository is acceptable, since the purpose is to see your personal code.

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 text, math, or code from each other are not allowed and will be considered as academic dishonesty. There cannot be any written material exchange between classmates, but verbal discussions are acceptable. Discussing with the instructor, the teaching assistant, and classmates at open office periods to get clarification or to eliminate doubts are acceptable.

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 (60%)

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 $[\pm 1, \pm 1, \pm 1]^T$. Pick four of these vertices as the mean vectors of the class-conditional Gaussian pdfs. For each class conditional Gaussian, also select an arbitrary non-diagonal covariance matrices with eigenvalues near the interval $[0.5, 1.4]$ in order to have some significant level of overlap between the tails of these Gaussian pdfs to make this question interesting.

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., ISRU, 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 10-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 (40%)

For this question use the *generateMultiringDataset.m* function to sample traning and testing data from class conditional distributions n the form of concentric rings; class priors are uniform. Generate a two-class training set with 1000 and testing set with 10000 samples. Train and evaluate a support vector machine classifier with a Gaussian kernel (radial-basis function (RBF) kernel) on these datasets. Specifically, use a spherically symmetric Gaussian/RBF kernel.

Using 10-fold cross-validation, select the best box constraint hyperparameter C and the Gaussian kernel width parameter σ (notation based on previously covered math in class from the SVM tutorial). Use minimum-average-cross-validation-probability-of-error to select best hyperparameters. Train a final SVM using the best combination of hyperparameters with the entire training set. Classify the testing dataset samples with this trained SVM to assess performance; estimate the probability error using the test set. Demonstrate numerical and visual results and explain how you trained and evaluated your SVM 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.