ECE 443 / ECE 539 (Fall'24) – Programming Exercise #3 (45 points)

Last updated: October 20, 2024

Rationale and learning expectations: This exercise reinforces key machine learning concepts related to probabilistic classification, focusing on plug-in classifiers such as Maximum Likelihood (ML), Maximum A Posteriori (MAP), and the plug-in Bayes classifier for a general loss function. Using real-world-inspired data on noise levels, students will train and evaluate these classifiers, applying machine learning techniques to classification tasks. The Gaussian assumption for noise levels allows for parameter estimation using maximum likelihood, while the general loss matrix highlights the importance of risk-sensitive classification. By the end of this exercise, students will understand how these classifiers minimize different types of loss and how empirical performance may deviate from theoretical guarantees.

General Instruction: All parts of this exercise must be done within a Notebook, with text answers (and other discussion) provided in text cells and commented code provided in code cells. Please refer to the solution template for Exercise #3 as a template for your own solution. In particular:

- Replace any text in the text cells enclosed within square brackets (e.g., `[Your answer to 1.1a goes in this cell]`) with your own text using Markdown and/or LaTeX .
- Replace any text in the code cells enclosed within pairs of three hash symbols (e.g., `### Your code for 1.1a goes in this cell ###`) with your own code.
- Unless expressly permitted in the template, **do not** edit parts of the template that are not within square brackets / three hash symbols and **do not** change the cell structure of the template.
- Make sure the submitted notebook is fully executed (i.e., submit the notebook only after a full run of all cells).

Restrictions: You are free to use `numpy`, `pandas`, `scipy.stats`, `matplotlib`, `mpl_toolkits.mplot3d`, `seaborn`, and `IPython.display` packages within your code. Unless explicitly permitted by the instructor, you are not allowed to use any other packages or modules.

Notebook Preamble: I suggest importing the different libraries / packages / modules in the preamble as follows (but you are allowed to use any other names of your liking):

```
import numpy as np
import pandas as pd
from scipy import stats as sps
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from IPython.display import display, Latex
```

1 Noise Level Classification Using Plug-in Classifiers

A team of environmental engineers is working on a system that monitors noise levels in various environments. To this end, they have collected data using a noise sensor that records the average noise level (in decibels, dB) over 5-minute intervals. The engineers aim to classify the environment into one of three categories based on the noise level:

- **Class 0:** Quiet environment (e.g., libraries, rural areas) with an average noise level around 45 dB.
- **Class 1:** Moderate noise environment (e.g., residential areas, offices) with an average noise level around 55 dB.
- **Class 2:** Noisy environment (e.g., urban areas, traffic) with an average noise level around 65 dB.

The data has been collected and processed by the engineering team. This means that tasks such as data cleaning and outlier removal have already been completed. The processed data is stored in two files:

- `NoiseClassificationTrainingData.csv`: Contains 400 data samples for training the classifiers.
- `NoiseClassificationTestData.csv`: Contains 100 data samples for testing the classifiers.

Each file includes two columns: *NoiseLevel* (in dB) and *ClassLabel*, where *ClassLabel* denotes the true class of the environment. The goal is to train classifiers to automatically predict the class of a new noise level measurement.

1.1. **Maximum Likelihood Estimation (MLE) of the Parameters:** Let us assume that the *NoiseLevel* feature for each class is normally distributed. In this case, the likelihood functions for the three classes are uniquely determined by the mean and variance parameters, which are unknown. Furthermore, assume that all three normal (i.e., Gaussian) distributions share the same variance.

  (a) (5 points) Estimate the means for each class (Class 0, Class 1, and Class 2) and the common variance using Maximum Likelihood (ML). *You should only use the training data for this purpose.* Since you are assuming a common variance for all classes, the variance must be estimated using the combined data from all three classes, ensuring that the respective mean of each class is subtracted from the corresponding data samples before combining them. Be cautious when using library functions for variance estimation, as some may return the unbiased estimate by default instead of the ML estimate. Print the ML estimates for the means and the common variance.

  (b) (2 points) Investigate the `numpy.var()` function and determine whether, by default, it returns the MLE or the unbiased estimate of the variance. What parameter in the function call allows you to switch between these two types of estimates? Summarize your findings.

1.2. **Analyzing the Distributions of the *NoiseLevel* Feature:** Let us now use histograms of the training data to assess whether the distribution of the *NoiseLevel* feature for each class (Class 0, Class 1, and Class 2) indeed follows a Gaussian distribution, i.e., whether the likelihood functions for the three classes are indeed Gaussian.

  (a) (6 points) Create a single plot that contains histograms of the *NoiseLevel* feature for each class (Class 0, Class 1, and Class 2) based on the training data. Each class should have a distinct color. On top of each histogram, overlay the Gaussian probability density function (PDF) corresponding to that class, using the MLEs for the means and the common variance (estimated in the previous question). These Gaussian PDFs represent the estimated likelihood functions for each class. Make sure to label the axes and provide a legend that clearly identifies both the histogram and the Gaussian PDF for each class.

  (b) (1 point) Based on the histograms and the overlayed Gaussian PDFs, do you think the Gaussian distribution with shared variance but different means is a reasonable assumption for the *NoiseLevel* feature across the three classes? Justify your answer based on your visual analysis.

1.3. (3 points) **Estimating Prior Probabilities:** Using the training data, estimate the prior probabilities for each class (Class 0, Class 1, and Class 2) by calculating the proportion of samples that belong to each class. Print the estimated priors for each class.

1.4. (15 points) **Classifying the Test Data Samples:** In this question, you will build three plug-in classifiers and use them to classify each test sample based on the *NoiseLevel* feature. These classifiers will allow you to predict the environment class (Class 0, Class 1, or Class 2) for the test data. For each classifier, print the following:

  - How many samples were classified as Class 0, Class 1, and Class 2.
  - For each actual class (Class 0, Class 1, and Class 2), how many samples were correctly classified.

  You will implement the following classifiers:

  - **Maximum Likelihood (ML) Classifier:** Using the MLEs of the means and the shared variance for each class, construct the ML classifier. This classifier will assign each test sample to the class whose Gaussian distribution is most likely to have generated the given sample (i.e., the class with the highest likelihood).
  - **Maximum A Posteriori (MAP) Classifier:** Incorporate the prior probabilities of each class along with the Gaussian likelihoods to construct the MAP classifier. This classifier assigns each test sample to the class with the highest posterior probability (likelihood times prior).

- **Plug-in Bayes Classifier for a General Loss Function:** In real-world applications, different types of misclassification errors can lead to different consequences, which need to be reflected in the classifier's decision process. The provided loss matrix

$$\mathbf{L} = \begin{bmatrix} -1 & 2 & 4 \\ 2 & 0 & 4 \\ 4 & 4 & 0 \end{bmatrix}$$

  reflects the impact of classification decisions. In this matrix, correctly classifying a sample from Class 0 (Quiet environment) is rewarded with -1, indicating the importance of accurate classification in quiet environments. Misclassifying Class 0 as Class 2 (Noisy) incurs the highest loss (4 units), since this would represent a significant overestimation of the noise level. Similarly, misclassifying Class 1 as Class 2 also incurs a high loss (4 units) because this too overestimates the noise level, which could lead to unnecessary actions being taken. On the other hand, misclassifications between Class 0 and Class 1 (or vice versa) are assigned a moderate loss of 2 units, reflecting that these environments are closer in noise level and the consequences of misclassifying between them are less severe than misclassifying into Class 2 (Noisy). Construct the plug-in Bayes classifier to minimize the expected loss based on this loss matrix. This classifier assigns each test sample to the class that minimizes the expected loss.

1.5. **Empirical Risk Evaluation:** After classifying the test data using the ML, MAP, and Plug-in Bayes classifiers, we will now assess their empirical performance. For each classifier, compute the following performance metrics:

(a) (3 points) Compute the empirical misclassification error (also known as zero-one loss) for each classifier. This represents the proportion of test samples that were incorrectly classified, or equivalently, the empirical risk under the zero-one loss function.

(b) (3 points) Using the provided loss matrix

$$\mathbf{L} = \begin{bmatrix} -1 & 2 & 4 \\ 2 & 0 & 4 \\ 4 & 4 & 0 \end{bmatrix},$$

calculate the empirical risk (or average loss) for each classifier. This involves computing the average loss across all test samples based on the classifier's predictions and the losses defined in the matrix.

1.6. **Theoretical Guarantees vs. Empirical Results:** In this problem set, it is assumed that the likelihood functions for each class are normally distributed with class-specific means and a shared variance, and it turns out that this assumption is indeed true. Theoretically, different classifiers, such as ML, MAP, and plug-in Bayes with a general loss function, are designed to minimize different types of expected losses. Answer the following based on your understanding of probabilistic classification frameworks in machine learning:

(a) (1 point) Which classifier is theoretically designed to minimize the expected zero–one loss (misclassification error)?

(b) (1 point) Which classifier is theoretically designed to minimize the expected general loss, given a loss matrix such as the one used in this exercise?

(c) (2 points) Do the empirical results reported in the previous question align with these theoretical guarantees? Specify for which classifiers the empirical results match the theoretical expectations and for which they do not.

(d) (3 points) When the empirical results do not align with the theoretical guarantees, why do you think this occurs? How might your observations change if both the training and test data sample sizes were to become extremely large? Explain why this would affect the alignment between empirical results and theoretical expectations.