CDS 1020-1: Introduction to Computational Data Science
Homework 3

Classification or giving a label to something is one of the main processes performed by humans. Since the advent of computer science, it has become possible to teach machines to automate human tasks like classification. From identifying images of Covid-19 chest X-rays to labeling emails as spam, classification algorithms have widespread application. Using labeled data, classifiers can learn the underlying density function of input data, optimize its parameters, and subsequently generate categorical variables or class labels. Here we will explore Bayes' theorem, a method of parameterization, and a form of parametric generative classification.

Assigned: Monday, March 14th, 2023, 11:59pm
Deadline: Friday, March 28th, 2023, 11:59pm

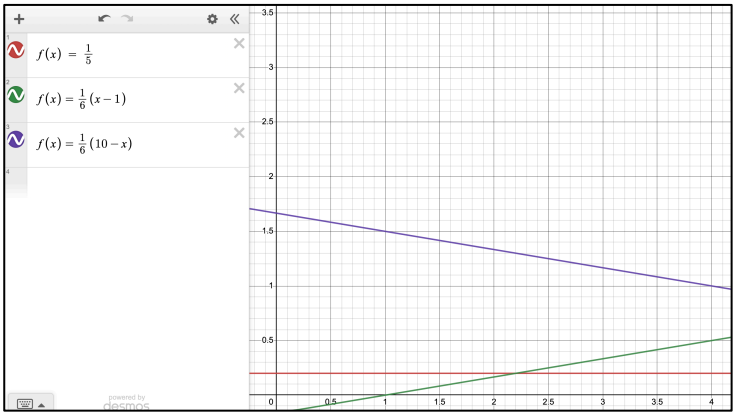
30 points

Part I: Constructing a Bayesian Framework (5 pts)

Using Bayes' theorem, we are interested in developing a pattern classifier. The input data is a single feature that is a continuous numerical variable; its values fall within the range $1 \leq x < 10$. The target variable is categorical, consisting of labels C_1 and C_2. The likelihood functions for each class, proportional to the conditional probability density function, are provided below. For classification, use the numerator of Bayes' theorem.

$$P(x|C_1) = \begin{cases} \frac{1}{5} & if\ 1 \leq x < 10 \\ 0 & otherwise \end{cases}$$

$$P(x|C_2) = \begin{cases} \frac{1}{6}(x-1) & if\ 2 \leq x < 6 \\ \frac{1}{6}(10-x) & if\ 6 \leq x < 9 \\ 0 & otherwise \end{cases}$$



Generated using Desmos

(a.) Assuming equal priors, $P(C_1) = P(C_2) = 0.8$, classify an object with the feature value $x = 8.9$.

(b.) Assuming unequal priors, $P(C_1) = 0.8$ and $P(C_2) = 0.3$, classify an object with the feature value $x = 3$.

Part II: Method of Parameterization—Maximum Likelihood Estimation (10 pts)
Machine learning algorithms are considered parametric if they assume a data distribution, corresponding to a known probability density function with specific parameters. Therefore, if one estimates the parameters using probabilistic methods like Maximum a Posteriori (MAP) or Maximum Likelihood Estimation (MLE), the distribution and thus behavior of the data can be understood. One algorithm known as logistic regression assumes linearity in the input data it learns, and a binomial distribution in the behavior of predictors to map them to target variables (with binary labels). By using MLE, logistic regression obtains the optimal model parameter or coefficient values that optimize its likelihood function, sufficiently fitting the data or predictors to target variables (i.e., performing binary classification).

Using Scikit-learn functionality, we are implementing a logistic regression model to classify diabetes status. The data is in the file Pima_Indians_diabetes.csv, and the script is PartII_HW3_CDS_1020.py. As you follow the script, write in code at the areas marked.

Part III: Programming Exercise—Developing and Implementing a Multivariate Gaussian Classifier (15 pts)
With its origins in statistics, linear discriminant analysis (LDA) is one of the oldest classification algorithms. By projecting labeled data points to a lower-dimensional space, LDA separates the data points according to their classes. With a modified assumption, quadratic discriminant analysis (QDA) can perform the same task. In this problem, we will use NumPy functionality to develop two multivariate Gaussian classifiers using LDA and then QDA. As we are performing binary classification, LDA and QDA will project to a dimensional space less than two (d = 1) while identifying unlabeled data.

- **Files:** two Jupyter notebook scripts. The “HW3_CDS_1020.ipynb” file contains the code for the models, and the “HW3_main_CDS_1020.ipynb” is the main file for generating the results of the classifiers.
- **Data:** The original data has been split into training and testing data files with the extension .txt: “training_dataset.txt” and “testing_dataset.txt.” Each data file contains a matrix $M \in \mathbb{R}^{NxD}$ where $N = 99$ samples and $D = 9$ dimensions or variables. In each data file, the first 8 columns are the features $X \in \mathbb{R}^8$, and the last column is the class labels $y \in \{1,2\}$.
- **Assumptions:** The two multivariate Gaussian classifiers using LDA and QDA will make the following assumptions, respectively:
    - Assume equal covariance matrices (i.e., S_1 and S_2 are learned independent of class; therefore, the calculation of covariance uses data from both classes. S_1 and S_2
    - Assume unequal covariance matrices (i.e., S_1 and S_2 depend on the data from each class).

1. In the definitions of the classes for both models (“GaussianDiscriminant_Dep and GaussianDiscriminant_Ind), what does the __init__ function do? In general, what

is another name for this function? In this script, what are its parameters, and what do they represent? Why are they not hard-coded (i.e., given values)? Why are the variables created here (e.g., mean, S) important for this type of model (i.e., how will they be used later in the class definitions for both models)? You may explain using words and/or using mathematical expressions.

2. *(Extra Credit, 3pts):* Generate the covariance matrix for the training data. What are the dimensions of this matrix? What does each cell of the matrix signify? Provide an explanation. Share your code.
3. What is the main purpose of a discriminant function in parametric generative classification? What is the discriminant function for LDA? Hint: the function is in the "HW3_CDS_1020" script and in Lecture 19.
4. What NumPy functions are used to code the discriminant function for LDA? Describe one: its parameters, the arguments it takes in the code, and what it generates.
5. How do the discriminant functions in LDA and QDA compare? What do they have in common with each other (i.e., distribution)? What do they have in contrast with each other; specifically, what is encoded in the discriminant function for QDA that is not encoded for LDA? Hint: consider their individual assumptions.
6. Deploy both models. Run "HW3_main_CDS_1020.ipynb" in script mode. This can be accomplished by converting the "HW3_CDS_1020.ipynb" Jupyter notebook to a python module, allowing classes defined here to be imported in the .py file like a python library. Hint: the output of the main script will be two confusion matrices. Share your code (i.e., the method you chose for converting the notebook to a python file) and output (i.e., the output of both classifiers).
7. What is the data structure of the object that contains the training and testing data prior to splitting further into X and y values ("dftrain" and "dftest")?
8. Interpret the results (i.e., the confusion matrices). Hint: examine the code creating the matrix underneath the "Evaluate model performance" section. How do the results of both classifiers compare with each other?
9. Implement both forms of discriminant analysis (i.e., LDA and QDA) with Scikit-learn. Compare the performance of the homemade algorithm for discriminant analysis to the performance of the automated, condensed Scikit-learn approach to discriminant analysis on the same training and testing datasets. If the homemade model is good, it will perform similarly to the models from the Scikit-learn library. Hint: use QuadraticDiscriminantAnalysis() and LinearDiscriminantAnalysis(). The only parameters required are the priors. Use the same priors as before for both classifiers. To evaluate the performance of your Scikit-learn implementations, you may replicate the previously used code for generating a confusion matrix. Share your code and output (i.e., the output of both Scikit-learn classifiers).
10. In the main script, call the class method get_parameters() for an object from both classes (i.e., the objects you generate when you instantiate the classes GaussianDiscriminant_Dep and GaussianDiscriminant_Ind). What is the output of this function? How do both outputs compare with each other in terms of dimensions? Provide an explanation for any differences. Share your code and output.

11. Run both homemade models for LDA and QDA on the alternative training and testing datasets. Provide possible explanations for any differences in confusion matrices between the original dataset (train and test together) and the alternative dataset (train and test together). There is no single right answer. Share code and output. Hint: compare dimensions of datasets, number of observations per class. What can you speculate about the distribution of each class? Does this affect performance?

Submission:
Submit a standard Python file (.py) for code, using the name "[last name]_Q[#]_HW3" to Canvas. Part II will be submitted as a Python file with original code in the designated areas. All coding portions for Part III will be submitted as a .py file. For visualizations and text answers, please use a document (.pdf) with appropriate numbering. You may scan your work for Part I. The submission to Canvas will consist of these three files.