CS 458 Assignment 1
Gaussian Discriminant Analysis

In this assignment, you will implement the Gaussian Discriminant Analysis-based Bayesian classifier to detect and segment the red barrel from the given images. All implementations must be done in Python3 language.

- Dataset labeling[2 pts]: We provide the training and testing dataset. Please use tools like Roipoly to hand-label appropriate regions in the training images with discrete color labels (https://github.com/jdoepfert/roipoly.py). We are particularly interested in parts containing the red barrel (positive examples) and parts that are not the red barrel (negative examples).
- Training [5 pts]: Use the hand-labeled data to train GDA parameters for binary classification. Implement two GDA algorithms, i.e., common variance and variable variance for the two classes.
- Image Segmentation [3 pts]: Use the trained GDA parameters to classify the colors on the test images. Replace image pixel values with red in which your classifier detected red color; for other regions, replace image pixel value with black color to form segmented images.
- Accuracy analysis [2 pts]: Use your trained model to determine the accuracy of your GDA models on the test datasets. Note that we will also test your models on a separate test dataset, not available to you, to validate the model performance. The accuracy of each GDA alrogithms can be demonstrated by a 2×2 table containing the average precision and recall of both labels (https://en.wikipedia.org/wiki/Precision_and_recall).

Programming instructions

Do not use any built-in functions that implement a core part of this project (Gaussian Discriminant Analysis). If you are not sure, then ask the TA. All python3 libraries allowed in this assignment are Numpy, math, roipoly, matplotlib, os, and sys.

You will need to implement the following functions in the provided template:

- label_training_dataset(training data path, "train_region"): This function should recursively go through all the image files in the given path and write your labeled positive and negative regions into files saved in a folder called "train_region". Make sure in each training image, you select the positive region and the negative region of similar sizes to avoid introducing bias to the model.
- label_testing_dataset(testing data path, "test_region"): Similar to the previous function, you need to manually label the region that contains the red barrel as the positive region and write them into files saved in a folder called "test_region". However, in this case, the negative region is just everywhere else.
- import_pre_labeled_training(training data path, "train_region"): This function loads your pre-labeled positive regions, negative regions as well as the training images and builds training features and labels.
- import_pre_labeled_testing(testing data path, "test_region"): This function loads your pre-labeled positive regions, negative regions as well as the testing images and builds testing features and labels.
- train_GDA_common_variance(features, labels): Fill this in with your implementation of GDA algorithm using a common covariance matrix. It should return priors, means, and a common covariance matrix.
- train_GDA_variable_variance(features, labels): Fill this in with your implementation of GDA algorithm using different variances for the two classes. It should return priors, means, and two unique covariance matrices.
- predict(features, priors, means, covariance matrices): Give each feature vector a predicted class label using your trained GDA parameters. This function should be designed to handle both GDA algorithms by checking the inputs.
- accuracy_analysis(predicted labels, ground truth labels): Implement this function to get the overall precision and recall for both classes. Print the numbers in the console with indicators of their meanings. Read the comments in the code template.

  Programming suggestions:

  - Please read the comments in the provided code template.
  - Please remember not to change the provided main function in your submission (but you will need to change it during the implementation, more details in the comments) as that is what the grading script is based upon. You are responsible for the grading issue caused by this.
  - The console output should only have 8 numbers: 2 precision(pos + neg) + 2 recall numbers(pos + neg) for each of the GDA algorithms with indicators of their meanings. Do not print out other unnecessary numbers in the console.

Submission instructions

The assignment is of 15 points: 12 points for the implementation + 3 points for the report. Your submission should consist of six items:

- Code template that contains all your implementations. Make sure this file is executable using "python3 CS458_Assignment_1.py" and the final accuracy values match with what your present in the report.
- Images.
  - train_region: this folder should contain all your labeled regions of the training data for the import function to load
  - test_region: this folder should contain all your labeled regions of the testing data for the import function to load
  - segmentation_GDA_common: this folder should contain all the segmented results of testing data from your trained GDA algorithm using a common variance
  - segmentation_GDA_variable: this folder should contain all the segmented results of testing data from your trained GDA algorithm using variable variances
- A PDF report in IEEE Latex template [3pts]. This report should contain

  Introduction: discuss why the problem is important and present a brief overview of your approach.

  Problem Formulation: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in.

  Technical Approach: describe your approach to barrel segmentation.

  Results: present your training results, and testing results followed by a discussion(what worked, what did not, and why). Make sure this section includes 3 segmented images as well as the original ones (1 from training and 2 from testing) and two 2×2 tables for both GDA algorithms reporting the accuracy of your models.

Note: if you have any special explanations of how your code works, or instructions for running it, feel free to place them either in the report or in a separate README file. Bundle all files into a zip or tarfile, and submit it via Brightspace. Name your submission "<first name_last name>.zip" or "<first name_last name>.tar.gz". For example, I would name my submission "Zixing_Wang.zip".