**ECE 8527: Introduction to  
Machine Learning and Pattern Recognition**

# HW No. 2: Bayesian DEcision Theory

For this assignment, you will use the data set located here:

*https://www.isip.piconepress.com/courses/temple/ece\_8527/resources/data/set\_04/*

We will focus on the files *train.txt* and *dev.txt*, which containing training and development test set data. You can assume the loss function weighs all errors equally.

The tasks to be accomplished in this homework assignment are:

1. Classify the data in dev.txt using a maximum likelihood classifier (assume the priors are equal). Compute the “experimental” error rate by classifying each point in the training and dev sets and scoring them against the true class assignment.
2. Assume the priors are not equal. Compute and plot the error rate as you vary the prior of class “0” over the range [0,1].
3. For the case where the priors are equal, compute the theoretical error rate that can be achieved on this data assuming you model each distribution as a multivariate Gaussian, and compare this to your experimental result.

# Exercise 1

In this exercise, a specific class is written to handle the training and the classification of the maximum likelihood classifier. The class, MaxLikeHood, takes in the training data and obtains the mean and denormalized covariance of the features as well as calculate the constant term in the form of a logarithm of the classification function during the initial setup. The subfunction, classify, carries out the remaining calculations in logarithm for the classification function that involves using the input array. The computed result is by default returned in the form of log probability. The classifier is trained by a specific set of data, train data in this case, and then verified with two sets of data. In the function, the error rate is also computed once the classification via the classifier is done and the result is printed in the Terminal window.

In this specific case, data from train and dev are read in separately. The classifier is trained via data from the train file and then verified with data from train and dev assuming the priors are equal. Correspondingly, the error rate from the train data is 49.57% and from the dev data is 49.57%. The data set that we used to train the classifier consist of 9000 input of each class with each input has 26 different features. We considered that 9000 inputs are sufficiently enough to train the classifier for this particular data set. The error rate obtained via the classifier is within the range of expectation as by reviewing the details of these features, some of them are not very distinguishable as found in the result of the previous homework exercise.

# Exercises 2

In this part of the exercise, the same classifier is taken into a loop that inputs a variety of the prior for the classification. The selected prior ranges from 0 to 1 with custom selected steps of 100. With each step of the prior, an error rate is computed with the classifier and stored in an array waiting to be plotted against the step of the prior. With all steps completed, a graph is generated showing the relationship between different priors and error rates of the classifier. In this specific case, the plot is generated using the prior probability of class 0.



Figure Class 0 Error Rate over Prior

The plot above shows how the error rate changes when the prior of class 0 changes. From the graph, it is very distinguishable that when the prior is at 0 or 1, the error rate is at 50% and there is a big improvement in the error rate when the prior is set to somewhere in between 0.6 and 0.7. It also shows that in a majority of the time, the error rate is stable at about 49.5%. By plotting this graph, we can find a relative prior rate to improve the detection rate of the classifier, namely in this case, set the prior to somewhere in between 0.6 and 0.7 in order to decrease the error rate of the classifier.

# Exercise 3

In order to compute the theoretical error rate of the function, the data that goes into the function needs to be whitened. A class, TheoreticalClassifier, is created for the purpose of whitening the data using PCA and with its subfunction, classify, to classify the data.

The first step of whitening the data is to obtain the zero-mean data of the class. The class is initialized with steps that obtain original mean and covariance of the data and then subtract the mean from each data to obtain the zero-mean data. The original covariance is then used to generate the eigen values and vectors with the aim of preparing the necessary elements of the data transform. The eigen values, eigen vectors, and the zero-mean data is then used to generate the whitened data with the function suggested in the lecture slide. A new variance is computed in order to verify that it is an identity matrix. The classifier is running by calculating the distance of the input vector to each feature points and comparing the distance to each class’ mean to make an educated guess of which class this specific vector belongs to.

During the exercise, it came to my attention that the covariance of the whitened data is quite close to the identity matrix with all other terms that is supposed to be 0 has a very small number in the scale of and the mean of the zero-mean data has the same behavior of its mean has values in the scale of . The result of the theoretical classifier is 49.95% for dev data and 50.6% for train data.