SFWR TECH 4DA3 Course Project
Comparing Classifiers

Introduction

In this course, we have looked at four different approaches to classifying data:

- Fishers Linear Discriminant
- Linear Support Vector Machine

Additionally, we have used these approaches in assignments where the implementation was provided by some code written in Python.

A general rule in machine learning is the "no free lunch" (NFL) theorem. Basically, this theorem states (from http://dml.cs.byu.edu/~cgc/docs/mldm_tools/Reading/LCG.pdf) that "*generalization is a zero-sum enterprise – for every performance gain in some subclass of learning situations there is an equal and opposite effect in others. As a result, the only way to determine which learning algorithm to employ for a specific problem is to try a number of algorithms and see which one works the best*". That is exactly what you are going to do in this project. As you gain experience with these algorithms when you apply them in your working careers, you will begin to see that some approaches work better than others for certain problems (subject, of course, to the NFL theorem). Thus you can make a more experienced choice of an algorithm which appropriately fits the problem at hand. Unfortunately, however, as is pointed out in

(http://web.archive.org/web/20140111060917/http://engr.case.edu/ray_soumya/eecs440_fall13/lack_of_a_priori_distinctions_wolpert.pdf) "Even after the observation of the frequent conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience." (from David Hume, in ***A Treatise*** of *Human Nature,* Book I, part ***3,*** Section 12).

all you can really do is make an educated guess of what algorithm will work well for your problem. This project will get you started down the path of making educated guesses!

Project Objective

In this project, your primary task is to **apply these 2 approaches** to a larger scale data classification task. In doing so, you will choose a **single data set** which you will classify with each approach. The data set must have greater than a **thousand** labeled examples. These examples must be classified into **one of two classes**. Note that it is possible to find a large data set which has more than two classes from which you could always select examples from only two of the classes. If a dataset contains mixed data (numeric and other types) you can always simply ignore the non-numeric data.

Once the data set has been classified, your job is to **compare the results** from the classifiers that you chose. The comparison (in general – details will follow below) must include the following analyses:

- Computational Times for both training and testing
- A confusion matrix

Project Methodology

Due to the size of the data set, you must use a **computer based implementation** to solve the problem. For this, you must use **Python** code. You can use either the code that was given to you for the SVM or for FLD or you can use the implementation in sklearn. You can do any data preparation or data post-processing (including graphing) in another software package such as Excel, but the core algorithm must be provided by Python.

Choosing a Dataset

You are free to choose any dataset that you like, subject to the constraint of **data types** mentioned below. The only other constraint is that the dataset must include at least **1000 labeled examples for two classes**. One excellent source of data is the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/index.php). These datasets are very popular amongst **researchers** in machine learning.

Training the Classifier

You must use at least 750 data points to train your classifier (the remaining 250 or more will be used for testing). In performing the training there are four analyses that you must perform:

Computational Time

To get a sense of how long it takes to train a classifier using different approaches with a moderate amount of data, measure the amount of time it takes to perform the training and testing. This measurement can be built into your code using a Python time function or you can simply record the time with a watch (to the nearest second is accurate enough) if the training or testing time is sufficiently long.

Recording the Results of Classification

A minimum of **25 percent** of your dataset must be reserved for testing. Note that this data is NOT to be used during the training of any of the algorithms including performing cross validation when determining parameters. The testing results of the algorithms must be shown using a confusion matrix (see below).

A confusion matrix is simply an indication of the errors resulting from classification. Note that are, in general, 2 types of errors that can be made – called **Type 1** and **Type 2** errors. Type 1 errors are when class B is instead classified as class A. Type 2 errors are when Class A is instead classified as class B. In the context that error types are defined this way, one of the classes, (class A for our example) is defined as a **positive** result. The other class is then defined as a **negative** result. Note that you can do this **arbitrarily** if you find that your two classes are **equally important** but for some datasets that you chose you may find that makes sense to define a **positive** and **negative** result. In any case, the confusion matrix is of the form:

Class B **misclassified** as Class A (**False positive** or Type 1 error)	Class A **correctly** classified (**True positive**)
Class B **correctly** classified (**True negative**)	Class A **misclassified** as Class B (**False negative** or Type 2 error)

In this matrix, you can record either the number in each category or a percentage. Note that the sum of elements in the first column must equal the number of elements in class B and the sum of elements in the second column must equal the number of elements in class A.

Project Submission

Individual submission is required. You cannot work in groups for this project – simply because the scope and time requirements are relatively small. Your submission must contain the following (and will be assessed based on the marks allocated for each part):

- Computational Times for both training and testing (5 marks)
- A confusion matrix (10 marks)
- Submission of Python code for all algorithms (5 marks)
- A two page (approximately) summary of all of the data that you provided in your write-up including how you obtained the results that you submitted. This should include what dataset you used, how you organized the data, how many training and test points you used, etc. (10 marks).

Your submission is due **the day of the final exam**. No extensions can be given, since this is close to the last day that I am permitted to submit the marks. Please upload your submission as a .zip file to the dropbox on the course website.