COMP TECH 4DM3 Course Project
Comparing Classifiers

Introduction

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

- K Nearest Neighbors
- Decision Trees
- Naïve Bayes
- AdaBoost
- Logistic Regression
- 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 3 of these 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.

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

- Computational Complexities (approximate) for both training and testing
- Cross Validation for parameter selection
- ROC (Receiver Operating Characteristic) curves
- 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 the **Python** code that has been provided with the course textbook – Machine Learning in Action. 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/). These datasets are very popular amongst **researchers** in machine learning.

Choosing a Classifier

As mentioned above, you must use 3 different approaches to classify data. In the course notes, we found that some approaches are more amenable to numeric data than to categorical data. While it is possible to modify all of the approaches to work with most types of data, we did not show these modifications in the course content. Therefore, the easiest thing for you to do is to choose 3 approaches that can easily handle the same type of data. The following table shows the types of data that we used to illustrate each of the 6 classification approaches.

Classification Method	**Data Type Used**
K Nearest Neighbors	**Numeric and Categorical**
Decision Trees	**Categorical**
Naïve Bayes	**Numeric and Categorical**
AdaBoost	**Numeric**
Logistic Regression	**Numeric**
Support Vector Machine	**Numeric**

This gives you some flexibility for which approaches you want to use. If you are interested in classifying **categorical** data, you should choose the **first 3 approaches**. If you wish to observe the behavior of more **modern classifiers** such as AdaBoost and the Support Vector Machine, you should find a data set that is primarily **numerical** in nature. If you wish to classify **mixed data**, I would suggest that you look into how to **modify Decision Trees** to handle numeric data. Note that even if a data set has attributes that are of **mixed type**, you can always simply **ignore attributes** that are of a type that you cannot handle with your classifier.

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. 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).

Training Parameters

Two of the algorithms – K Nearest Neighbor and the Support Vector Machine (with a Radial Basis Function kernel) have parameters that must be adjusted to provide reasonable results. In this project **you must use at least one of these 2 techniques and use cross-validation to determine the parameter**. For K Nearest Neighbor, the parameter to be determined is the number of nearest neighbors used, or K. For the Support Vector Machine, the parameter is σ, the standard deviation of the kernel.

We will discuss the details of cross-validation in the final lecture. In the meantime, please refer to the following document "**Cross Validation.pdf**" in the Content->Classifier Evaluation section of the course website for how to perform parameter selection with cross-validation. The basic idea is very simple. To perform cross-validation, simply divide the training set into pieces – testing on one pieces and training on the others. In general, we can perform what is called **d-fold cross validation**. The idea is that we divide the training data set into **d sets**, training on every possible combination of **d-1 sets** and **validating on the remaining set**. For example, with 750 data points used for training, we could perform 3-fold cross validation by randomly dividing the training data set up into 3 groups of 250 data elements each. We would then perform 3 experiments, each with a different 250 elements used for validation (and therefore 500 elements used for training). The 3 experiments would be repeated for each value of the parameter that is to be determined and the classification error for that value of the parameter would be averaged over the 3 experiments. Taken to the extreme, we can have **d=n** or 750-fold cross validation (commonly called leave-one-out (LOO) validation), where 750 experiments would be repeated for each value of the parameter that is to be determined. For this setup, only one data point is validated in each experiment. Regardless of the value of d that is chosen for the validation, the parameter is selected based on the value that produced the minimum average error during all of the experiments.

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 in two ways.

First, a confusion matrix must be calculated. This 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.

The second way that your results **must be shown** is using an ROC (Receiver Operating Characteristic) curve. This will be discussed in more detail in the final lecture. In the meantime, please refer to the document provided here ("**ROC.pdf**" in the Content->Classifier Evaluation section of the course website) for a description of an ROC curve. The idea is quite simple. Every classifier (with the exception of Decision Trees) has a score associated with the classification. The larger the score, the more likely it is to be in one class or the other. Conceptually, what is done to create an ROC curve is to adjust the threshold of the score used to separate the two classes and record the number of **True Positives** vs **False Positives** as a point on a graph at each value of threshold. This is computationally time consuming and messy to do. So what is normally done is that the data is sorted according to score from highest to lowest and then run down through list, updating number of True Positives vs False Positives as we go.

Note that a ROC cannot be shown (easily – one approach is described in ("**ROC with Decision Trees.pdf**" in the Content->Classifier Evaluation section of the course website)) for a Decision Tree based classifier so if you choose this approach, an ROC curve result is not necessary. Also note that the score is not as obvious for K Nearest Neighbors. However, provided that K is large enough, the score can simply be tabulated as the number of neighbors that agree in the majority vote. So, if K is 49 and 45 neighbors agree on Class A, than the score is 45/49 for Class A. This would apply similarly for Class B. Points which are near the decision boundary would have scores of 25/49 as Class A (24 out of 49 as Class B) and 24/49 as Class A (25/49 as Class B) respectively. With a small K, the number of points on the ROC curve would be small.

Project Submission

Due to the limited number of students in the class, each group must contain a **maximum of 2 students**. One submission is required for each group. 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)
- Cross Validation for parameter selection (10 marks)
- ROC (Receiver Operating Characteristic) curves (10 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 **on 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.