Title: Naive Bayes vs Logistic Regression
Author: Logan Turske
Phone: 989.590.0098

Abstract:

Some of the simplest models in machine learning are the linear models. In this experiment we implement two linear models that help to classify our data. Linear models tend to follow Occam's Razor pretty well and that is why we tend to learn and choose linear models first. There are some drawbacks to using the two linear models that we implemented, Naive Bayes and Logistic Regression, which we will explain in further detail below. Overall the results of the experiment coincided with our views that these linear models will classify the data well.

Problem Statement:

We wish to find the average classification accuracy for each of the five datasets {cancer.csv[1], glass.csv[2], iris.csv[3], soybean.csv[4], vote.csv[5] } using the linear model Naive Bayes and Logistic Regression.

Hypothesis:

I believe that given a dataset the Naive Bayes and Logistic Regression will have a classification accuracy greater than 50%.

Naive Bayes:

Naive Bayes is a technique that allows different flavors of algorithms to create classifiers. One of the major assumptions that Naive Bayes makes is that each feature vector of the data set is independent of one another. The algorithm also makes the assumption that the data follows a distribution, which combined with the previous assumption can make for some very strong assumptions. Using, for example, a Gaussian distribution you update a model using the Gaussian probability equation for each data instance you are given. We summarize all of the features of the data and attempt to predict what class something belongs to. The algorithm is naive in the sense that the data points are independent of each other, as stated earlier, which means that we start to stray away from the Occam's Razor principle quickly when we are trying to decide which algorithm to use on our data. Bayesian classifiers assign the most likely class to a given example described by its feature vector [6].

Logistic Regression:

Logistic Regression is an algorithm that attempts to classify a data point based upon some feature vector weight. As we saw in this experiment, if we wish to use Logistic Regression in its purest form, we must using binary features along with a binary classification decision. There is a way to change your data to satisfy these conditions as we will explain below. One of the most important mathematical properties that logistic regression builds on is the logit function - the natural logarithm of an odds ratio[7]. We use this property to classify a data point compared to the weights we have seen so far. Defined in our Python script are two functions “softmax”, “sigmoid”, and “log_likelihood” which deal with determined the weights and classifications of the data points.

Experimental Approach:

In this experiment we had to deal with binary class and multiclass classification problems. The interesting part is how Logistic Regression deals with multiclass datasets as compared to Naive Bayes. To implement Logistic Regression in a multiclass problem we had to change the dataset using "one-hot coding". One-hot coding in short, changes your data so that the features of your data can be represented in some binary form. This causes the actual feature and class columns to grow very wide. Due to how wide the dataset gets, we have left the actual datasets unmodified so the reader can actually parse the data themselves. As you will see below[Table 1], the Logistic Regression algorithm and the Naive Bayes algorithm differ on performance depending on what data they are given.

Results:

Evaluation Metric: Classification error

Data	Naive Bayes	Logistic Regression	Class Problem
cancer.csv	94.3%	95.58%	binary
glass.csv	85.93%	89.9%	multi
iris.csv	97.51%	95.21%	multi
soybean.csv	4.9%	94%	multi
vote.csv	94.93%	96.53%	binary

Table 1.

Behavior of Algorithms:

In our opinion, the algorithms work in a very similar way. Even though the underlying math they use to determining weights is different, the behavior is still the same. We see from the table above [Table 1] that the difference in performance varies from dataset to dataset. We believe this is because of how the data is given to us and after using one-hot coding how the data loses some of accuracy because the weights may be thrown off. We can see this is in the Naive Bayes run against the soybean.csv[4] dataset.

Summary:

In summary, both of these linear algorithms perform fairly well against the datasets. We have proved our hypothesis to be true in that we expected the algorithms to have a classification accuracy greater than 50%, aside from one dataset using Naive Bayes. We found that linear models are easy to implement and they have very few assumptions for the most part. In conclusion, linear models should be used if you understand your data well and can manipulate the presentation of the data in a meaningful way.

References:

[1] William Wolberg. Breast Cancer Wisconsin. Retrieved April 15, 2018 from https://archive.ics.uci.edu/ml/datasets/breast cancer wisconsin (original)
[2] B. German, Glass Identification Data Set. Retrieved April 15, 2018 from https://archive.ics.uci.edu/ml/datasets/Glass+Identification

[3] R.A Fisher, Iris Data Set Retrieved April 15, 2018 from https://archive.ics.uci.edu/ml/datasets/Iris

[4]Michalski R.S, Soybean (Small) Data Set Retrieved April 15, 2018 from https://archive.ics.uci.edu/ml/datasets/Soybean+%28Small%29

[5]Jess Schlimmer, Congressional Voting Records Data Set, Retrieved April 15, 2018 from https://archive.ics.uci.edu/ml/datasets/Soybean+%28Small%29

[6] I. Rish, An Empirical Study of the Naive Bayes Classifier. 41-42

[7]Chao-Ying, An Introduction to Logistic Regression Analysis and Reporting, 9-10