Introduction to Machine Learning, Programming Assignment 5

Comparing Linear Classifiers: Logistic Regression vs. Naïve Bayes

Julie P. Garcia

Johns Hopkins University

Abstract

Linear classifiers are commonly used in making predictions in machine learning. In this method a classification is made on linearly separable data by learning a linear discriminant function to separate the two classes. In this paper, we evaluate two methods for linear classification by implementing two algorithms: Naïve Bayes and Logistic Regression. Naïve Bayes and Logistic Regression are similar methods in that they both can be used in classification problems using posterior probability, however the way that they learn are different in that they use posterior probability in different ways.[1,3]

Hypothesis

In this paper, we demonstrate that Naïve Bayes works very well on most data sets large and small, however as the number of features grow, the performance of Naïve Bayes degrades as a linear classifier. Logistic regression also performs as well as a linear classifier, however, performance degrades as the number of features increase, as well. Both algorithms only work well as a linear classifier on linearly separable data.

Algorithms Implemented

We implemented Logistic Regression and Naïve Bayes in order to compare their performance on a number of data sets. Both algorithms use the posterior probability to make predictions. Naïve Bayes uses a model of the distributions of the features and the class values within the features, while logistic regression models the posterior probability by creating a mapping between input and output.

Logistic Regression Algorithm

Logistic Regression has a basis in probabilistic methods. It uses the logic distribution to find the optimal weights of each feature. We apply gradient descent to the algorithm by continually updating the weights with the weight delta.[1,4] The algorithm was implemented as follows:

1. Initialize an array of weights with the size equal to the number of features in the dataset to random values in the range (-0.01, 0.01).
2. Loop until the weight vector converges or a max number of iterations is executed.

   a. Set weight deltas as a vector of the same length as the weights and initialize them to zero
   b. Loop through all of the examples of the data set
      i. Sum the feature value times the weight
      ii. Pass result through the logistic function sigmoid to determine the class prediction
      iii. Compare real class value with class prediction and calculate the weight delta
   c. Calculate the gradient and update the weight vector

Naïve Bayes Algorithm

The Naïve Bayes algorithm uses a probabilistic model to classify samples of data. Naïve Bayes only works well for linearly separable data sets, therefore, when using this algorithm, we introduce the inductive bias that the data in our data set is linearly separable.[3] The algorithm was implemented as follows:

1. The model is constructed by finding P(C=0) and P(C=1) in dataset, where C is the classifier, and then P(F=1|C=0), P(F=0|C=0), P(F=1|C=1) and P(F=0|C=1) for each sample in the dataset.
2. Predictions are made my by iterating through each sample in the dataset and multiplying P(C=0) by the probabilities for each feature in the sample to get the total probability for that combination, given C=0 and given C=1. If the P(C=0) > P(C=1) the prediction is 0, otherwise it is 1.

Experimental Approach

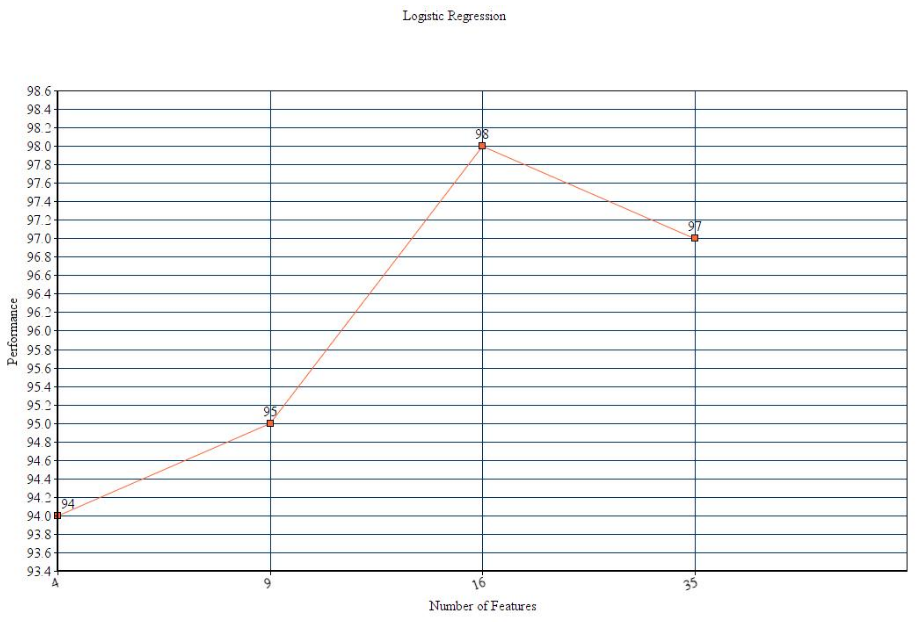
All algorithms were run on five classification datasets downloaded from the UCI Machine Learning Repository: Breast Cancer, Glass, Iris, and Soybean.[2] For this a stratified 5-fold cross-validation method was used. Each dataset was divided into five randomly-selected equal datasets. The datasets are stratified to ensure equal representation of classes across each set. Five experiments were run, each time selecting a different partition (20%) as the test set and the remaining 80% as the training set. This allows for each of the 5 sets to rotate as the test set. Averages were calculated over the five experiments.

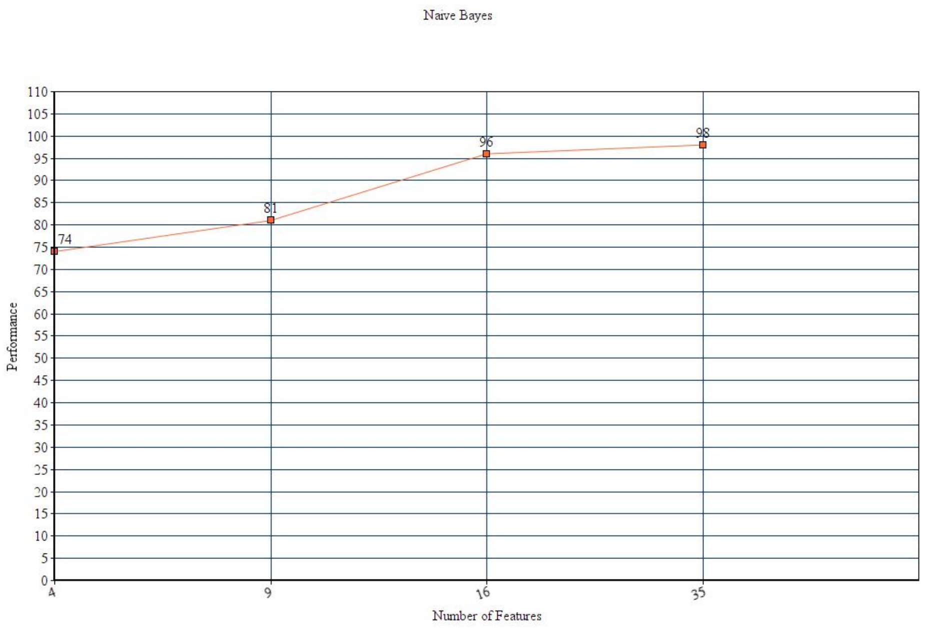
Because we are using logistic regression as a linear classifier, the weights did not converge very easily, so we set a max iterations on the algorithm and stopped when it hit the maximum number of iterations. This value was tweaked and the optimal value of 50 was set that maximized the accuracy of the model.

Results

When run on the Breast dataset (683 examples, 9 features) both algorithms performed very well with ~95% accuracy for logistic regression and ~81% accuracy for Naïve Bayes. On the Soybean dataset (47 examples, 35 features), again, both algorithms performed very well with ~98% for logistic regression and ~97% for Naïve Bayes. For the Votes dataset (435 examples, 16 features) both algorithms performed well with ~96% accuracy for logistic regression and ~98% for Naïve Bayes. The Iris dataset, with the least number of classes (150 examples, 4 classes) had good performance with Naïve Bayes ~97%, but logistic regression did not perform as well with ~74% accuracy. Finally, for the Glass dataset (214 examples, 9 classes) both algorithms performed very poorly at ~67% for logistic regression and ~33% for Naïve Bayes.

Below is a chart showing the four datasets that performed fairly well (Iris, Breast, Soybean, and Votes) and their performance. This shows how performance increases as the number of features increase for both algorithms.





Summary

In summary, the Naïve Bayes algorithm works very well as a linear classifier on data sets that have linearly separable data. The performance of Naïve Bayes degrades as the number of features grow. Logistic regression also works well as a linear classifier on linearly separable data, however performance seems to degrade as the number of classes increase.

Based on these results I suspect the first four data sets (Breast, Soybean, Iris, and Votes) have linearly separable data and that the Glass data set does not. In the future, I would do more analysis the Glass dataset to see if a more complicated model is necessary.

References

1. Alpaydin, E. (2014). *Introduction to machine learning*. Cambridge, MA: The MIT Press.
2. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
3. Osbourne, J. W. (2014). "Multinomial and Ordinal Logistic Regression." *Best Practices in Logistic Regression*, pp. 388–433., doi:10.4135/9781483399041.n12.
4. Vapnik, Vladimir Naumovich. *Statistical Learning Theory*. Wiley, 1998.