Naïve Bayes vs Logistic Regression (CS605.449 – Project 5)

By Max Robinson

Abstract

In this experiment the performance of logistic regression classification on five datasets is analyzed and compared to the performance of a Naïve Bayes classifier. The models for each were built and tested on five datasets, Breast Cancer, Glass, Iris, Soybean, and Vote from the UC Irvine Machine Learning repository. The results of the performance showed that the logistic regression classifier performed equally or better on all datasets, except for the Breast Cancer dataset. The largest performance gains for logistic regression classification were on the more complex dataset, Glass, where there was a decrease of error rate of around 20% compared to Naïve Bayes classification on a few of the classes.

Problem

The problem being investigated in this experiment is how well the Naïve Bayes classification algorithm performs on five different datasets when compared to a logistic regression classification algorithm. The five datasets used are the Breast Cancer, Glass, Iris, Soybean, and Vote datasets from the UC Irvine Machine Learning Repository.

For this experiment, I hypothesize that the logistic regression classifier will perform as well or better than the Naïve Bayes classifier on all five datasets. While both Naïve Bayes and Logistic Regression are linear models, Naïve Bayes classification assumes that all features are independent of all other features given the class for any data point. This assumption is simplistic and is often not accurately reflected in the data, though it can perform surprisingly well in domains where this assumption does not hold true [1].

Logistic regression does not make this assumption about the features in the datasets. As such I predict that it will perform better, especially on datasets that have data with missing features or more complex data, such as the data in the glass dataset.

Algorithm Implementation

Naïve Bayes Classifier

The Naïve Bayes classifier works by essentially a "database" of records, from which the distribution of the data is reduced. Specifically, the values calculated are $P(f|c)$ for some feature value for a given feature with respect to some classification label. In doing this, we are assuming that each feature value is independent of every other feature value, given the class. The Naïve Bayes classifier has been shown to be optimal in reducing the misclassification rate or error rate, given that the assumption that all variables are independent of each other given the class [1]. In addition, however, it has been shown empirically to perform well in domains with attribute dependencies [1].

For this experiment, we transformed all of the data into binary feature sets so each feature is referenced by feature number (i.e. feature 0,1,2,3 etc). The given values that are possible for any given feature number or $f = 0 \; or \; 1$, since we are using binary features. Since we are also using binary classes, positive or negative class, the possible values for classes are $c = 0 \; or \; 1$. As such the possible probabilities found for a given feature are $P(0|0), P(0|1), P(1|0), P(1|1)$.

Each of these probabilities are calculated from the training set. When a new feature vector needs to be classified, the probabilities for each feature value are looked up for each class, and multiplied together. This value is then normalized for each class based on the probability of that class in the entire dataset. The resulting classification is the class that has the highest probability for that class.

Formally this means that given an example E, and a discriminant functions $f_i(E)$ that correspond to the ith class, the chosen class C_k is the class where the following is true, $f_k(E) > f_i(E) \; \forall i \,!= k$.[1]

For this specific implementation, we also used a +1 smoothing tactic to ensure that no probabilities for a given feature were zero. If any probabilities were zero, this can cause the entire probability to be zero, which would provide often incorrect predictions, or no prediction at all for a class.

Logistic Regression with Gradient Descent

Logistic Regression (LR) classification works as a linear discriminant based classification. The goal of logistic regression is to find a hyperplane defined by $\boldsymbol{W}$ with a threshold of w_0 such that the hyperplane bifurcates the test and training data into the two corresponding classes of the dataset. This assumes that the dataset being used to train this model is of two classes. In the case of multiple classes in the dataset, a one-vs-one or one-vs-all approach is often taken.

The following equation is used to describe $\boldsymbol{W}$ such that $\boldsymbol{W}$ optimally bifurcates the data.

$$\boldsymbol{W} = argmax_w \prod_k P\left(C^k \middle| \boldsymbol{X}^i, \boldsymbol{W}\right)$$

$$\boldsymbol{W} = argmax_{\boldsymbol{W}} \sum_k ln \; P\left(C^k \middle| \boldsymbol{X}^i, \boldsymbol{W}\right)$$

$$l(\boldsymbol{W}) = \sum_k \left[C^k \left(w_0 + \sum_{j=1}^{d} w_j x_j^k \right) - \ln \left(1 + exp\left(\sum_{j=1}^{d} w_j x_j^k \right) \right) \right]$$

Since this equation has no closed form, it cannot be solved directly. As such, the method of gradient descent is applied to this equation to maximize the equation and find the argmax for $\boldsymbol{W}$.

To classify a data point using a model the following equation is used.

$Choose \; \{C_1 \; if \; g(\boldsymbol{X}) > 0, C_2 \; otherwise\}$ which is equivalent to $Choose \; C_1 \; if \; \boldsymbol{W}^T\boldsymbol{X} > -w_0$

w_0 is stored as part of the model as well as the other members of $\boldsymbol{W}$, to allow for these equations to be easily used for classification.

Logistic regression does not have the same limitations as Naïve Bayes classification since it does not rely on the fact that all features must be independent of each other given the class. As such this linear model is typically able to generalize better and be applied to datasets that have dependent features given the class.

Gradient Descent is a method for function minimization that was originally proposed by Cauchy [2]. For this experiment we specifically used steepest descent which is often synonymous with gradient descent [2]. The idea behind gradient descent is that if one takes a step along the negative gradient of a function, $f(x)$, then the value of the function should decrease. Continued steps along the negative gradient of the function should continue to decrease the function value until a local or global minimum is reached.

For this experiment, the function that gradient descent is being applied to is the derivative of the $l(\boldsymbol{W})$ function described above which can be written as follows.

$$\frac{\delta l(\boldsymbol{W})}{\delta w_i} = 0 = \sum_k x_i^k \left(C^k - \hat{p}\left(C^k = 1 \,\middle|\, \boldsymbol{W}\right)\right)$$

To update a parameter w_i then the following equation is used.

$$w_i = w_i + \alpha \sum_k x_i^k \left(C^k - \hat{p}\left(C^k = 1 \,\middle|\, \boldsymbol{W}\right)\right)$$

In this update equation, α determines the size of the step that is taken along the gradient. The typically method used for steepest descent does a search for alpha as follows $\alpha_k = \ argmin_\alpha \ f(x_k + \alpha d_k)$ where $d_k = -\nabla f(x_k)$ [2]. What this describes is finding alpha such that the $f(x_k)$ along the negative gradient is the smallest value possible, thus finding the maximal step length to take along the gradient.

For this experiment, instead of doing this search for α_k a fixed value of α is used for the step size to take along the negative gradient. This alpha only ever changes should a step down the gradient increase the error term $\left(C^k - \hat{p}\left(C^k = 1 \,\middle|\, \boldsymbol{W}\right)\right)$ to a larger value than previous. This ensures that steps down the gradient to not start to climb out of the function local minimum. This can occur if the step size is too large. For this experiment $\alpha = \ .1$.

For gradient descent, the $\boldsymbol{W}$ continues to be updated until some convergence criteria is met. For this experiment, convergence is met once all $w_i \ \epsilon \ \boldsymbol{W}$ have stopped changing by less than some value ϵ. This means that for convergence $\forall w_i \ \epsilon \ \boldsymbol{W}, \ \ |w_k - \ w_i| < \epsilon$, where w_k are the previous values of $\boldsymbol{W}$. For this experiment $\epsilon = \ .0001$.

This value constitutes as a meta parameter for the gradient descent algorithm and can be tuned according to the results and error rates. For this experiment, epsilon was chosen after anecdotal experimentation across a subset of the test datasets.

Scoring

Error Rate

Error rate is used in this experiment to calculate how many misclassifications occurred when compared to the total number of data points classified. Error rate can be described as

$$errorRate = \frac{\#\ of\ misclassified\ test\ instances}{total\ number\ of\ test\ instances}$$

Experimental Approach

The approach for this experiment was two fold. The first part was done through data preprocessing, and the second part was done through how the experiments were run. The metric used for comparison of the two algorithms in this experiment was error rate, which is simply the number of incorrect predictions divided by the total number of items in the test set.

Data Pre-processing

Each data set was pre-processed to turn all of the features into binary features, 0's and 1's. The process for each data set was a little different, since they all had different ranges of values and types of values.

For the Breast Cancer data set, the ID feature was removed from the set since it was a unique id and does not provide correlation with the data. The remaining features all have values between 1 and 10 so nine new features were created for each one and a one's hot encoding was applied to each new feature. As a result, there were 90 total features, plus the class label per feature vector.

For the Glass data set, the unique id was removed. All values were floats, and had a max and min range for the data set. These values were then binned into 6 bins. A one's hot encoding was applied to which bin a given value fell into.

The iris dataset was also real numbers based and had a maximum and minim value for each feature. The same tactic of binning and one's hot encoding was used for this dataset. 6 bins were also used.

The soybean dataset was integer based, with a maximum value in the entire data set of 6. As such, each integer feature was given 6 binary features, and a one's hot encoding was applied. For missing features, a random value between 0 and six was generated and used in its place.

The house votes dataset was binary based, so all the values were transposed into 1's and 0's. For missing values, a random number 0 or 1 was generated for that place. This data set had the largest portion of missing data.

Cross Validation and Validation Set

For this experiment, cross validation was used to provide robust and averaged results across different distributions of the data. Five-fold cross validation was used for this experiment. This means that from the complete set of the data, five partitions are created. For each run of Logistic Regression or Naïve Bayes the five folds are rotated through, were four of the folds are merged to create training set of 80% of the data, and one fold is used as the test set, 20%. The partition that is used for the test set is rotated so that every partition is used as the test set once.

The data used for each fold in the cross validation are also stratified since all data sets were classification problems. This means that the distribution of the data for each fold mirrors that of the distribution of the data in the entire training set. If class A makes up 40% of the data in the entire data set, then class A will make up 40% of the data in each fold for cross validation.

Classification

To get comparable results, each algorithm was run on each data set with a one-vs-all classification approach with cross validation. As such, each data set was tested with each possible class as the positive class.

For data sets with more than two classes, this means that the positive class used was selected at run time, and one class was selected. All other records that were of a different class were then lumped into the negative class. Each possible class value for each dataset was used as the positive class. This allows error rate to be measured for each class individually for the data set.

Error rate for each algorithm was calculated for each fold. The error rates were then averaged across the number of folds, in this case five folds. The standard deviation between the folds was the n also calculated.

Results

After the running the experiment as described in the above sections, the average error rates for each dataset and each positive class can be viewed in Table 1. The table lays out each data set on each row, with sub rows describing which class was used as the positive class the data set. The average error rates for each algorithm are in adjacent columns for ease of comparison.

Table 1 Average Error Rate of Logistic Regression and Naïve Bayes per Dataset and Positive Class

Error Rate Table	Positive Class Used	Logistic Regression Average Error Rate	Naïve Bayes Average Error Rate	Logistic Regression SD	Naïve Bayes SD
Breast Cancer	Benign	0.043478	0.027536	0.012126	0.007800
	Malignant	0.042029	0.028986	0.026086	0.012126
Glass	1	0.302439	0.346341	0.066528	0.066169
	2	0.365854	0.424390	0.051161	0.042526
	3	0.087805	0.302439	0.011949	0.045237
	5	0.039024	0.180488	0.011949	0.073333
	6	0.024390	0.234146	0.0	0.076509
	7	0.043902	0.073170	0.028444	0.034493
Iris	Iris Setosa	0.0	0.0	0.0	0.0
	Iris Versicolour	0.073333	0.106667	0.024944	0.057349
	Iris Virginica	0.066667	0.106667	0.047140	0.024944
Soybean (small)	D1	0.0	0.1	0.0	0.093541
	D2	0.0	0.0	0.0	0.0
	D3	0.05	0.075	0.061237	0.099999
	D4	0.025	0.025	0.05	0.05
House-Votes	Democrat	0.062791	0.1	0.011858	0.013953
	Republican	0.065116	0.102326	0.005696	0.015426

Behavior

The behavior of the two algorithms and the corresponding error rates were as expected for all datasets except for the Breast Cancer dataset.

When comparing the Logistic Regression results to the Naïve Bayes results, the differences in average error rate ran the gambit. The largest gain in performance for the logistic regression algorithm was a decreased error rate of .214634 or about a 21.4% decrease in error rate on the glass dataset for class 3.

This is a difference of almost 18 standard deviations difference using the logistic regression standard deviations. It is about 4.7 standard deviations away using the Naïve Bayes standard deviations. This I very large difference and points at the logistic regression classifier being consisten tly and statistically more accurate virtually all the time. A similar example is class 6 for the glass dataset with very similar differences in error rate and distance in terms of standard deviations.

On other classes in the glass dataset, the logistic regression average error was sometimes withi n one or two standard deviations of the Naïve Bayes classifier. Examples are classes 1, 2, and 7. I believe this points toward characteristics in the data and the differences between the different classes that allowed for such a wide range in improvements. For classes where logistic regression performed much better than Naïve Bayes, I think it is due to the complex nature of the data and the inability for Naïve Bayes to learn the corresponding relationships between the features. The glass dataset had real valued features that were modified to be bucketed one's hot encodings. I think that the logistic regression algorithm lost less information from the transformation of the data.

For the Iris, Soybean datasets, logistic regression and Naïve Bayes generally performed within one or two standard deviation of each other. Logistic regression did perform better for all classes for both datasets though. This is as expected. However, for a few of the classes the algorithms performed either the same, or had zero error. I believe that this is likely due to the fact that both datasets are relatively small. With a smaller dataset, it is possible that the data is not diverse enough, or the linear regression model overfit to the data.

The votes data was interesting in that logistic regression performed more than three standard deviations better for the democratic class and about 6.5 standard deviations better on the republican class using the logistic regression standard deviation. The voting dataset was the dataset with the most noise in the dataset. As a result, it seems that the logistic regression algorithm was better able to handle the noise than the Naïve Bayes classifier.

Defying expectations was the Breast Cancer dataset. For the benign class the difference of the average error was about .015942. This is about two standard deviations away using the Naïve Bayes standard deviation, and about 1.3 standard deviations away using the logistic regression standard deviation. The malignant class was similar but closer, with a difference of 0.13043 in error rate and a distance of about 1 and .5 standard deviations away with Naïve Bayes and logistic regression standard deviations respectively. While this shows that the performance of the two algorithms are close, the Naïve Bayes classifier is likely to perform better than logistic regression.

I believe that this shows an interesting thing about the Breast Cancer data. The data seems to show that based on the way the features were decomposed in the data translation into one's hot encodings for each feature, the features seem to be fairly independent of each other given the class. This would explain why Naïve Bayes performs so well to begin with. This is the scenario where Naïve Bayes performs best since the algorithm is optimal for when this assumption is true [1].

Another reason that logistic regression might have performed worse than Naïve Bayes for this algorithm is due to the chosen epsilon convergence value for gradient descent. Since after translating the data, there were 90 features per data point, the value for epsilon might have needed to be much smaller to reach a better stopping point for the model that was more accurate.

Summary

Overall, the algorithms performed as expected when comparing the results of the Naïve Bayes classifier and the logistic regression classifier, except for once dataset. The increases in performance of the logistic regression classifier was scattered but across most of the datasets, the logistic regression classifier outperformed the Naïve Bayes classifier. On the glass dataset, which is arguably the most complex, logistic regression classification had some of its largest gains in performance. For simpler datasets, the gains were modest or none. The single outlier was the Breast Cancer dataset where the Naïve Bayes classifier outperformed the logistic regression classifier by a moderate amount.

References

[1] Domingos, Pedro, and Michael Pazzani. "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss." Machine Learning, vol. 29, no. 2, 1 Nov. 1997, pp. 103–130., doi:10.1023/a:1007413511361.

[2] Meza, J. C. (2010), Steepest descent. WIREs Comp Stat, 2: 719–722. doi:10.1002/wics.117

Data Sources

Breast Cancer — https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29
Glass — https://archive.ics .uci.edu/ml/datasets/Glass+Identification
Iris — https://archive.ics.uci.edu/ml/datasets/Iris
Soybean (small) — https://archive.ics.uci.edu/ml/datasets/Soybean+%28Small%29
Vote — https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records