Journal of Machine Learning Research 1 (2019) 1-10 Submitted 9/19; Published 9/19

CSCI 447: Project 1

George Engel GeoEngel.z@gmail.com
Department of Engineering
Montana State University
Bozemane, MT 59715, USA

Troy Oster toster1011@gmail.com
Department of Engineering
Montana State University
Bozeman, MT 59715, USA

Dana Parker danaharmonparker@gmail.com
Department of Engineering
Montana State University
Bozeman, MT 59715, USA

Henry Soule hsoule427@gmail.com
Department of Engineering
Montana State University
Bozeman, MT 59715, USA

Editor: Engel et al.

Abstract

A brief, one paragraph abstract summarizing the results of the experiments

Keywords: Keywords, Go, Here

1. Introduction

The naive Bayes classification model is a simple model for predicting data classification. In this assignment, we implemented this probabilistic model on multiple real-world datasets. We validated our implementation of the algorithm with 10-fold cross-validation and measured the performance of our implementation with two loss functions: 0-1 Loss, and Precision/Recall.

2. Problem Statement

Given a data set of which all the correct classifications are known, we can implement and execute a training algorithm on a subset of the data. Then, once trained, the algorithm will guess the classes of another subset of the data and measure the performance. For this assignment, we were provided with 5 real-world data sets. Our task was to implement a training algorithm and guess classifications for each of these 5 data sets and then utilize two different loss functions to measure the accuracy and performance of our predictions.

3. The Algorithm

Per the instructions of the assignment, we implemented the naive Bayes algorithm on the 5 provided data sets. Given some example $x \in X$, where X is our data set, naive Bayes predicts the correct class c of x by computing the probabilities of each possible classification for x. For class c, the probability is denoted as $P(c|a_1, a_2, ..., a_d)$ where a_k denotes one of d attribute values in x. To compute this probability for each class c, the probabilities of each attribute value are computed. For each attribute value a_k, we compute $P(c) * \prod_{i=0}^{d} P(a_i|c)$, where $P(c)$ is the probability of an attribute being classified as class c. To predict the correct class for x we compute $\underset{c \in C}{\text{argmax}}\ P(c) * \prod_{i=0}^{d} P(a_i|c)$.

4. Our Approach

To properly implement naive Bayes on the 5 data sets, we first needed to properly separate and classify each data set. Each data set needed to be separated by class, and then the count and probability of each attribute value for each class needed to be computed. Our approach for this was to create, for each dataset, an associative array, where each possible classifier was a key. For each key, the corresponding values were each their own associative arrays, the keys of which were all the extant attribute values among that class. The value of each key in these inner arrays was a set storing the count and conditional probability of each attribute value given the respective class.

We tested our algorithm's implementation using 10-fold cross-validation. This validation model separates each data set into 10 subsets, or "bins", of equal size. It then runs 10 iterations. During each iteration, one bin is designated as the test data, being the data we will test our prediction algorithm on. The other 9 bins are designated as training data. It is on this training data subset that we will execute our learning algorithm, from which we will base our predictions. After first running the training algorithm on these 9 bins, we then predict the classifications of each example in the designated test data. For each iteration, a different bin is designated as test data, while the other 9 are designated as the training data.

To assess the performance of our class prediction algorithm, we implemented two loss functions: Precision/Recall, and 0-1 Loss. We ran both loss functions on each iteration of the 10-fold cross-validation process, and then took averages of the values returned during from each iteration and used these values to measure the performance of our prediction algorithm. 0-1 Loss simply computes the fraction of the data our algorithm correctly classified. For each correctly classified data example, a 0 is returned, and for every incorrectly classified value, a 1 is returned. Accuracy is then defined as the percentage of 0's returned [2].

The Precision/Recall loss function computes two values, known as precision and recall, to measure the performance of our class prediction algorithm. This loss function utilizes the amounts of true positive, false positive, and false negative classifications for each class.

Precision and recall are are computed as follows:

$$Precision = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FP_i}$$

$$Recall = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FN_i}$$

TP_i denotes the number of true positive classifications for class i, FP_i denotes the number of false positive classifications for class i, and FN_i denotes the number of false negative classifications for class i. C is the set of all possible classes. Precision can be interpreted as a measure of how accurate true positive classifications are, as it computes the fraction of all positive classifications for a class that were truly positive. Recall measures accuracy among the values for each class that should have been positive, as it computes the fraction of all values that truly belonged to a class that were classified as positive [1].

5. Results

Acknowledgments

1. Davis, Jesse, and Mark Goadrich. "The Relationship between Precision-Recall and ROC Curves." Proceedings of the 23rd International Conference on Machine Learning - ICML 06, 2006, doi:10.1145/1143844.1143874.

2. Dietterich, Thomas G. "Machine Learning for Sequential Data: A Review." Lecture Notes in Computer Science Structural, Syntactic, and Statistical Pattern Recognition, 2002, pp. 15-30., doi:10.1007/3-540-70659-32.

Appendix A.

In this appendix we prove the following theorem from Section 6.2:
Theorem *Let u, v, w be discrete variables such that v, w do not co-occur with u (i.e., $u \neq 0 \;\Rightarrow\; v = w = 0$ in a given dataset $\mathcal{D}$). Let N_{v0}, N_{w0} be the number of data points for which $v = 0, w = 0$ respectively, and let I_{uv}, I_{uw} be the respective empirical mutual information values based on the sample $\mathcal{D}$. Then*

$$N_{v0} \;>\; N_{w0} \;\;\Rightarrow\;\; I_{uv} \;\leq\; I_{uw}$$

with equality only if u is identically 0. ■
Proof. We use the notation:

$$P_v(i) \;=\; \frac{N_v^i}{N}, \quad i \neq 0; \quad P_{v0} \;\equiv\; P_v(0) \;=\; 1 - \sum_{i \neq 0} P_v(i).$$

These values represent the (empirical) probabilities of v taking value $i \neq 0$ and 0 respectively. Entropies will be denoted by H. We aim to show that $\frac{\partial I_{uv}}{\partial P_{v0}} < 0$....

Remainder omitted in this sample. See http://www.jmlr.org/papers/ for full paper.