CSE537 Homework 5

Problem 1: Clickstream Mining with Decision Trees

Problem 2: Spam Filter Using Naive Bayes

2017 Fall Semester

Group 29

Name: Haotian Wang Student ID: 111496343 Email: haotwang@cs.stonybrook.edu
Name: Richard Sicoli Student ID: 109181138 Email: richard.sicoli@stonybrook.edu
Name: Balraj Inder Kaur Mahal Student ID: 111498697 Email: balrajinderka.mahal@stonybrook.edu
Name: Shree Nath Dutt Sharma Student ID: 111425141 Email: richard.sicoli@stonybrook.edu

1 Introduction

This project involves mining click-stream data collected from Gazelle.com. We have to determine the visitor will view another page on the sites or leave, given a set of page views. We need to implement ID3 Decision Tree to classify these visitors, according to 247 attributes from training data set.

2 ID3 Decision Tree

A Decision Tree represents a function that takes as input a vector of attribute values and returns a "decision" — a single output value. The input and output values can be discrete or continuous. For now we will concentrate on problems where the inputs have discrete values and the output has exactly two possible values; this is Boolean classification, where each example input be classified as true (a positive example) or false (a negative example).

At each level, we should choose attribute to classify the data. The greedy search used in decision tree learning is designed to approximately minimize the depth of the final tree. The idea is to pick the attribute that goes as far as possible toward providing an exact classification of the examples. A perfect attribute divides the examples into sets, each of which are all positive or all negative and thus will be leaves of the tree.

We use the notion of information gain, which is defined in terms of **entropy**. In general, the entropy of a random variable V with values v_k, each with probability $P(v_k)$, is defined as

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

Every time, we choose the best attribute with max information gain under each subtree.

3 Chi Squared Test

The Decision Tree Learning algorithm will generate a large tree when there is no pattern to found, which is called overfitting. In general, overfitting occurs with all types of learning. Thus, we implement Chi Squared pruning to combat overfitting. It works by eliminating nodes that are not clearly relevant.

We need to calculate the probability that, under the null hypothesis, a sample of size $v = n + p$ would exhibit the observed deviation from the expected distribution of positive and negative examples. We can measure the deviation by comparing the actual numbers of positive and negative examples in each subset, p_k and n_k, with the expected numbers, $\hat{p}_k$ and $\hat{n}_k$, assuming true irrelevance:

$$\hat{p}_k = p \times \frac{p_k + n_k}{p + n}$$

$$\hat{n}_k = n \times \frac{p_k + n_k}{p + n}$$

A convenient measure of the total deviation is given by

$$\triangle = \sum_{k=1}^{d} \frac{(p_k - \hat{p}_k)^2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)^2}{\hat{n}_k}$$

Under the null hypothesis, the value of $\triangle$ is distributed according to the chi-squared distribution with $v - 1$ degrees of freedom. We can use a particular $\triangle$ value confirms or rejects the null hypothesis.

We make the decision tree algorithm stop generating nodes when there is no good attribute to split on, rather than going to all the trouble of generating nodes and then pruning them away. The problem with early stopping is that it stops us from recognizing situations where there is no one good attribute.

4 Simulation

We run the test cases provided by the website. We set the threshold for the chi-square criterion set to 0.01, 0.05, and 1. When the threshold is 1, we will get the full decision tree. Our accuracies and tree size of each test are demonstrated below:

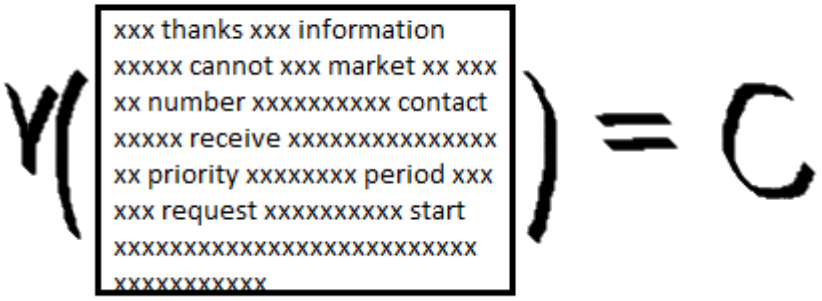
Threshold	Accuracy	Size
0.01	0.730	541 nodes
0.05	0.727	1866 nodes
1	0.710	27226 nodes

We can see the accuracies are good but still have much room to improve. We also see that lower chi-square thresholds significantly reduce the tree size. We can use C4.5 Decision Tree algorithm to improve ID3 Decision Tree algorithm. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain. The attribute with the highest normalized information gain is chosen to make the decision.

Using C4.5 Decision Tree algorithm can handle continuous attributes. Because we know the training data and test data are partitioned into 5 intervals of equal frequency, which generates some error. In addition, C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes. In this way, the accuracies can be improved.

5 Naive Bayes Classification

Naive Bayes classifiers are a popular statistical technique of e-mail filtering. They typically use bag of words features to identify spam e-mail, an approach commonly used in text classification.



Naive Bayes classifier is very good in domains with many equally important features. In contrast, decision trees suffer from fragmentation in such cases, especially if little data.

5.1 Multinomial naïve Bayes

This is the event model typically used for document classification, with events representing the occurrence of a word in a single document. In a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, p_2, ...p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $(x_1, x_2...., x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance.The likelihood of observing a histogram x is given by:

$$p(x|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

5.2 Implementation

For our problem we can set S as the probability of the given email being spam, and C as the contents of the email, which are given to us in a bag of words model. For each email, if the conditional probability of an email being spam given the contents is greater than the conditional probability of it not being spam given the contents, we can classify the email as spam, otherwise we cannot. As we have seen in above sections that the denominators are same in both of the probabilities, we can just compare the numerators to get our decision.

5.2.1 Calculation of Individual probabilities

Calculation of P(spam) is trivial given our labeled dataset, and similar is the P(not spam), we can just get counts of total spam and not spam emails from the training dataset.

5.2.2 Calculation of conditional probabilities

Since we will be using a bag of words model, we can just calculate the fraction of times a word occurs in a spam, and a non spam email. This is our conditional probability - P(word|spam) and P(word|not spam).

5.2.3 Testing

Now that we have got individual word probabilities, we just need to compute the same for each word in the given test email, and multiply them together to get the probability of an email being spam. This will be computed in two phases. We will compute the conditional probability of a collection of words, given they were present in a spam email, and multiply them with the individual probability of an email being spam. This will give us the probability of an email being spam. Similarly, we will compute the probabilities for an email not being spam and compare the two. Although, depending on the estimate, we can vary the fraction, but for sake of simplicity we just compare if Probability of being spam is greater than that of not being a spam, and usedd that for our decision.

5.2.4 Optimization

Up until now, since we are relying on bag of words model, if we encounter a word out of our dictionary, we do not have any prediction about the word, so we have to set up the probability to zero. However, We can add a smoothing to out multinomial distribution, by adding an alpha value as the frequency of occurrence, when we see a new word that is not in our dataset. This is called Laplace Smoothing, and using this generally helps in an increase in accuracy.

6 Results

We were able to achieve an accuracy of 0.765 with the basic naive bayes based spam classification implementation on the provided test dataset. However, with a different test dataset than the training, the implementation would not work since any zero probability would disturb the overall probability function. Hence, we then implemented Laplace Smoothing and were able to get an increase in accuracy of 0.79

Following are the results with varying values of alpha -

Algorithm	Alpha	Accuracy
Multinomial Naive Bayes	-	76.5
Multinomial Naive Bayes with Laplace Smoothing	1	76.9
−− same −−	10	76.3
−− same −−	100	76.0

7 Contribution

We are group 29. At first, all four of us studied the materials of this project and discussed about these two problems. Richard.Sicoli and Haotian.Wang focused on problem 1: Clickstream Mining with Decision Trees. Balraj Mahal and Shree Nath focused on Problem 2: Spam Filter. Balraj implemented the initial part of the problem formulation, i.e. creating the bag of words for spam and non spam emails from the dataset. Shree Nath, then implemented the Bayes conditional Probabilities for each word, eventually using them to calculate the probability of the whole email being a spam or not, and later the laplace smoothing. After getting this done, Balraj and Shree Nath both worked on separate parts of the report about Problem 2. After each subgroup finished their program and report, two group share their program and report to double check.