

# **PROJECT 05: MACHINE LEARNING**

## **ARTIFICIAL INTELLIGENCE**

### **CSE 537**

#### **INTRODUCTION**

##### **Clickstream Mining with Decision Trees**

The project is based on a task posed in KDD Cup 2000. It involves mining click-stream data collected from Gazelle.com, which sells legwear products. Your task is to determine: Given a set of page views, will the visitor view another page on the site, or will he leave?

The data set given to you is in a different form than the original. It has discretized numerical values obtained by partitioning them into 5 intervals of equal frequency. This way, we get a data set where for each feature, we have a finite set of values. These values are mapped to integers, so that the data is easier for you to handle.

##### **DATASET**

have 5 files in .csv format

1. `trainfeat.csv`: Contains 40000 examples, each with 274 features in the form of a 40000 x 274 matrix.
2. `trainlabs.csv`: Contains the labels (class) for each training example (did the visitor view another page?)
3. `testfeat.csv`: Contains 25000 examples, each with 274 features in the form of a 25000 x 274 matrix.
4. `testlabs.csv`: Contains the labels (class) for each testing example.
5. `featnames.csv`: Contains the "names" of the features. These might useful if you need to know what the features represent.

##### **Stopping Criterion: Chi-squared criterion**

A chi-square ( $\chi^2$ ) statistic is a test that measures how expectations compare to actual observed data (or model results). The data used in calculating a chi-square statistic must be random, raw, mutually exclusive, drawn from independent variables, and drawn from a large enough sample.

Suppose that splitting on attribute T, will produce sets  $\{T_i\}$  where  $i = 1$  to  $m$

$$p'_i = p \frac{|T_i|}{N}$$

$$n'_i = n \frac{|T_i|}{N}$$

be the expected number of positives and negatives in each partition, if the attribute is irrelevant to the class. Then the statistic of interest is:

$$S = \sum_{i=1}^m \left( \frac{(p'_i - p_i)^2}{p'_i} + \frac{(n'_i - n_i)^2}{n'_i} \right)$$

Let  $p$ ,  $n$  denote the number of positive and negative examples that we have in our dataset (not the whole set, the remaining one that we work on at this node). Let ( $N$  is the total number of examples in the current dataset):

There are two main kinds of chi-square tests: the test of independence, which asks a question of relationship, such as, "Is there a relationship between gender and SAT scores?"; and the goodness-of-fit test, which asks something like "If a coin is tossed 100 times, will it come up heads 50 times and tails 50 times?"

For these tests, degrees of freedom are utilized to determine if a certain null hypothesis can be rejected based on the total number of variables and samples within the experiment.

For example, when considering students and course choice, a sample size of 30 or 40 students is likely not large enough to generate significant data. Getting the same or similar results from a study using a sample size of 400 or 500 students is more valid.

In another example, consider tossing a coin 100 times. The expected result of tossing a fair coin 100 times is that heads will come up 50 times and tails will

come up 50 times. The actual result might be that heads comes up 45 times and tails comes up 55 times. The chi-square statistic shows any discrepancies between the expected results and the actual results.

## **OBSERVATIONS**

For  $p = 0.01$

Internal nodes: 26 Leaf nodes: 105

For  $p = 0.05$

Internal nodes: 35 Leaf nodes: 141

For  $p = 1$

Internal nodes: 265 Leaf nodes: 1061

## **ACCURACY**

Tree Prediction Accuracy: 0.75216

Output file prediction Accuracy: 0.75216

## **SPAM FILTER**

The dataset we will be using is a subset of 2005 TREC Public Spam Corpus. It contains a training set and a test set. Both files use the same format: each line represents the space-delimited properties of an email, with the first one being the email ID, the second one being whether it is a spam or ham (non-spam), and the rest are words and their occurrence numbers in this email. In preprocessing, non-word characters have been removed, and features selected similar to what Mehran Sahami did in his [original paper](#) using Naive Bayes to classify spams.

## **NAIVES BAYES ALGORITHM**

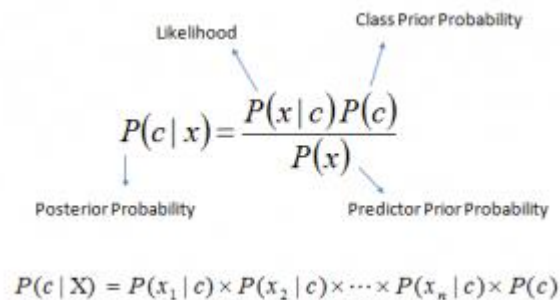
It is a [classification technique](#) based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes

that the presence of a feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability  $P(c|x)$  from  $P(c)$ ,  $P(x)$  and  $P(x|c)$ . Look at the equation below:



The diagram shows the equation  $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$  with arrows pointing from labels to the terms: 'Likelihood' points to  $P(x|c)$ , 'Class Prior Probability' points to  $P(c)$ , 'Posterior Probability' points to  $P(c|x)$ , and 'Predictor Prior Probability' points to  $P(x)$ . Below this, the joint probability equation is given:  $P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$ .

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Posterior Probability      Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Above,

- $P(c|x)$  is the posterior probability of class ( $c$ , target) given predictor ( $x$ , attributes).
- $P(c)$  is the prior probability of class.
- $P(x|c)$  is the likelihood which is the probability of predictor given class.
- $P(x)$  is the prior probability of predictor.

### OBSERVATION

Accuracy with smoothing: 93.3