Matt Forbes
February 2011
CS 400 - AI Indep. Study
Learning With Naive Bayes Classification

Overview

Situation: We want to classify some data into categories but there is no formal definition describing how this is done. Given some number of test cases, we can come up with an efficient way of classifying unknown instances. Ideally, we could use the Bayes optimal classifier algorithm, but it is computationally infeasible. Making some simplifying (usually incorrect) assumptions, we can devise an "easier" way of doing this classification.

If we let x be some instance composed of conjoined attributes $a_i \ldots a_n$, and V be the set of possible values an instance can be classified as, then the probability of x being classified as v_j is:

$$P(v_j|a_1, a_2, \ldots, a_n).$$

To find the most probable v_j, we take the argmax of the above probability. With Bayes rule, we can simplify this expression and obtain:

$$P(v_j)P(a_1, a_2, \ldots, a_n|v_j)$$

Up until this point, we have not made any assumptions, and this expression would indeed give us the most probable v_j for this instance. However, coming up with these probabilities is completely hopeless.

If we were so lucky that each a_i was conditionally independent of the others, we would be in business. From rules of probability we know that $P(a, b|c) = P(a|c)P(b|c)$ when a and b are conditionally independent given c. Using this, we can express the most probable v_j as:

$$\underset{v_j \in V}{\arg\max}\ P(v_j) \prod_{i=1}^{n} P(a_i|v_j)$$

This whole situation just got much easier computationally. $P(v_j)$ is easy; it is just $\frac{1}{|V|}$ unless we know some values are inherently more probable. Each value of $P(a_i|v_j)$ can also be calculated fairly inexpensively. Based on the training data, this would be the number of times a_i occurred in instances classified as v_j divided by the total number of these v_j instances.

Only a small subset of problems will satisfy this constraint of conditionally independent instance attributes. It is claimed, though, that in practice making this incorrect assumption will still yield very reasonable results, making this technique very appealing because of its simplicity. Because the claim of conditional independence is usually incorrect, this technique is called naive Bayes classification.

Text Classification

Described in chapter 6 of Mitchell is a simple process used to classify text documents using the above result. A statement of our problem is this: we have 20,000 text documents all pertaining to some given set of categories. If classification of 18,000 documents is known, how do we classify the remaining 2,000?

I implemented naive Bayes classification with guidance from Michell's description of such an algorithm. Each document instance is considered as a conjunction of its words with no regard to position. This last condition, of not considering position, is the simplifying assumption that allows us to use conditional independence in calculating probabilities.

If we let C be the set of possible categories we can classify a document as, and some document d a conjunction of n words w_i, then using naive Bayes we find the most likely category for d as:

$$\arg\max_{c_j \in C} \; P(c_j) \prod_{i=1}^{n} P(w_i | c_j)$$

$P(c_j)$ is the number of documents known to be classified as c_j divided by the total number of documents. Denoting the set of all documents as X and the relation d "classified as" c_j to be $d \in c_j$ then the above expression becomes:

$$\arg\max_{c_j \in C} \left(\frac{|\{x : x \in X, x \in c_j\}|}{|X|} \right) \prod_{i=1}^{n} P(w_i | c_j)$$

Now, the function $occurrences(w, c)$ returns the number of times the word w has occurred in all documents previously classified as c. Also, the function $wordcount(c)$ returns the total number of unique words within all documents classified as c. Using these two function we can represent $P(w_i|c_j)$ simply as:

$$P(w_i | c_j) = \frac{occurrences(w_i, c_j)}{wordcount(c_j)}$$

Our expression for the most likely classification of d is as follows:

$$\arg\max_{c_j \in C} \left(\frac{|\{x : x \in X, x \in c_j\}|}{|X|} \right) \prod_{i=1}^{n} \frac{occurrences(w_i, c_j)}{wordcount(c_j)}$$

In my final implementation, I used an m-estimate probability technique recommended by Mitchell to increase the accuracy of the term $P(w_i|c_j)$. This estimate is most useful when the size of the training data is small and some examples can drastically skew the results of the rest of the classification process. Because we have 20,000 well-distributed training documents, this estimate doesn't have much influence.

Results

First off, a little about the data used. I used 20,000 emails collected from 20 different newsgroups. These were provided by Mitchell from his CMU website. From each category, I removed 200 documents to be used for testing. So, my program was trained on 18,000 documents, then classified the remaining 2,000 based on what it learned.

A general sketch of the algorithm is as follows:

1) Observe the training data directory and build the set of categories C.
2) Create a hash-table called vocab that will be built from unique words seen in all the documents
3) For each each category, build a hash-table that will hold word frequencies. Read through every document in this category word-by-word to calculate a frequency for each word. Training is complete.
4) Try classifying the remaining documents in the "classify" directory using the naive bayes classification algorithm discussed above. When correctly classified, increment a counter.
5) Return the number of documents classified correctly out of the total attempts.

When run against the data, my program yielded 1707/2000 documents classified correctly, roughly 85% accuracy.