Ways to improve the accuracy of a Naive Bayes Classifier?

34

I am using a Naive Bayes Classifier to categorize several thousand documents into 30 different categories. I have implemented a Naive Bayes Classifier, and with some feature selection (mostly filtering useless words), I've gotten about a 30% test accuracy, with 45% training accuracy. This is significantly better than random, but I want it to be better.

25

I've tried implementing AdaBoost with NB, but it does not appear to give appreciably better results (the literature seems split on this, some papers say AdaBoost with NB doesn't give better results, others do). Do you know of any other extensions to NB that may possibly give better accuracy?

Thanks very much.

machine-learning bayesian

share improve this question

edited Apr 24 '14 at 9:39
manlio
9,027 10 26 43

asked Aug 13 '10 at 3:06
wmute
338 1 4 8

Use a Bayesian Network Classifier instead of a Naive Bayes Classifier. – George Aug 13 '10 at 9:42

add a comment

2 Answers

active oldest votes

67

In my experience, properly trained Naive Bayes classifiers are usually astonishingly accurate (and very fast to train--noticeably faster than any classifier-builder i have everused).

so when you want to improve classifier prediction, you can look in several places:

- *tune your classifier* (adjusting the classifier's tunable paramaters);
- apply some sort of *classifier combination technique* (eg, ensembling, boosting, bagging); or you can
- look at *the data* fed to the classifier--either add more data, improve your basic parsing, or refine the features you select from the data.

w/r/t naive Bayesian classifiers, parameter tuning is limited; i recommend to focus on your data--ie, the quality of your pre-processing and the feature selection.

I. Data Parsing (pre-processing)

i assume your raw data is something like a string of raw text for each data point, which by a series of processing steps you transform each string into a structured vector (1D array) for each data point such that each offset corresponds to one feature (usually a word) and the value in that offset corresponds to frequency.

- *stemming*: either manually or by using a stemming library? the popular open-source ones are Porter, Lancaster, and Snowball. So for instance, if you have the terms *programmer, program, progamming, programmed* in a given data point, a stemmer will reduce them to a single stem (probably *program*) so your term vector for that data point will have a value of 4 for the feature program, which is probably what you want.
- *synonym finding*: same idea as stemming--fold related words into a single word; so a synonym finder can identify developer, programmer, coder, and software engineer and roll them into a

single term

- *neutral words*: words with similar frequencies across classes make poor features

II. Feature Selection

consider a prototypical use case for NBCs: filtering spam; you can quickly see how it fails and just as quickly you can see how to improve it. For instance, above-average spam filters have nuanced features like: frequency of words in all caps, frequency of words in title, and the occurrence of exclamation point in the title. In addition, *the best features are often not single words but e.g., pairs of words, or larger word groups*.

III. Specific Classifier Optimizations

Instead of 30 classes use a **'one-against-many' scheme**--in other words, you begin with a two-class classifier (Class A and 'all else') then the results in the 'all else' class are returned to the algorithm for classification into Class B and 'all else', etc.

The Fisher Method (probably the most common way to optimize a Naive Bayes classifier.) To me, i think of Fisher as *normalizing* (more correctly, *standardizing*) the input probabilities An NBC uses the feature probabilities to construct a 'whole-document' probability. The Fisher Method calculates the probability of a category for *each* feature of the document then combines these feature probabilities and compares that combined probability with the probability of a random set of features.