A Gaussian Naive Bayes Classifier

Machine Learning (CS 545), Portland State University, Winter 2016

Bruce Marron

25 February 2016

Introduction

Naive Bayes classifiers are supervised learning algorithms that apply Bayes' theorem with the "naive" assumption of independence between every pair of features. That is, for a class variable y and a dependent feature vector x_1 through x_n,

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \textstyle\prod_{i=1}^{n} P(x_i \mid y)$$

A second common assumption when using Naive Bayes classifiers is that the likelihood of each feature is Gaussian; that is,

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

This report documents the use of a Gaussian Naive Bayes algorithm to classify email as 'spam' or 'not spam' given a feature vector in $\mathbb{R}^{57}$. The "Spambase" dataset was the source of raw data for generating as well as for evaluating the classifier. The Spambase data were obtained from the University of California, Irvine (UCI) Machine Learning Repository. All data processing and evaluation tasks were done in Python 2.7.11 (Anaconda 2.4.1 (32-bit)) using the integrated development environment (IDE), "Spyder". The open source, "scikit-learn" machine learning package was used as a reference for classifier code construction.

Data Processing

The raw data in Spambase contains 4601 cases of which 1813 are identified as 'spam' ('1' in Field 58) and 2788 are identified as 'not spam' ('0' in Field 58). The entire Spambase dataset was shuffled twice and then split into a training dataset (`tr_d`, n=2300) and a test dataset(`te_d`, n=2301). The prior probabilities for spam (=1) and not spam (=0) in the training dataset were given as,

$$\begin{aligned} P(y=1) &= 0.4070 \\ P(y=0) &= 0.5930 \end{aligned}$$

The mean and variance for each feature $(x_1, \ldots, x_{57})$ was calculated for the spam subset of `tr_d` as well as for the not-spam subset of `tr_d`. In instances where the variance of a feature was equal to zero, a proxy variance (=1e-10) was entered to prevent division by zero in the classifier. This resulted in a set of four, 1x57 vectors,

`tr_d_mu_spam`

`tr_d_sigma_spam`

`tr_d_mu_notspam`

`tr_d_sigma_notspam`

A Naive Bayes algorithm was then applied to the test dataset `te_d` and each case subsequently classified as either spam(=1) or not spam(=0). The procedures and methods used for all data processing tasks are detailed in the scripts, `DataProcessing1.py` and `DataProcessing2.py`.

Results and Discussion

The results of the Naive Bayes classification are presented in Table 1 and in Table 2.

Table 1: The accuracy, precision, and recall of a Naive Bayes classifier as applied to the Spambase dataset.

Classifier	Accuracy	Precision	Recall
Naive Bayes	0.7245	0.7008	0.4835

Table 2: The confusion matrix for a Naive Bayes classifier as applied to the Spambase dataset.

		Predicted	
		Spam	Not Spam
Actual	Spam	1243	181
	Not Spam	453	424

Conclusions

It is unlikely that the assumption of independence is valid for the features in the Spambase dataset because a ”spammer”, like any author, is expected to use and repeat certain phrases (symbol and word sequences) with high frequency. In spite of this, the classifier performed rather well. A more sophisticated version might keep the assumption of independence but relax the assumption of normality and use more feature-specific likelihood distributions.