Part II: Programming and Questions

Note: You must do your programming in Python.

Together with this homework, we have posted a training and test files for a simple spam classification problem. (Real-world spam classification tasks today are much more difficult!)

An example in this dataset corresponds to an email, and the attributes correspond to properties of the email. The problem is to classify the example into one of 2 classes: spam or not-spam. [1]

Files: The essential information about the data is given in the file `spambase1.txt`. There are 9 continuous input attributes, as described in the file. The class label is designated as 1 (spam) or 0 (non-spam).

The examples are given in two files:

the training examples are in `spambasetrain.csv`

the test examples are in `spambasetest.csv`.

Each line of these two files gives the information for one example. The first 9 columns have the values of the 9 input attributes, in the order specified in the `spambase1.txt` file. So, for example, column 3 contains the value for the attribute `char_freq_[`. The class of the example is given in the final column.

To do: Implement Gaussian Naive Bayes in Python.

Gaussian Naive Bayes is very similar to the standard Naive Bayes algorithm described in the written part of this homework (and in the Lecture 2 slides), but it is designed for datasets with numeric attributes. As in standard Naive Bayes, it uses the conditional independence assumption of Naive Bayes, which says that

$$p(x_1, x_2, \ldots, x_d|C) = p(x_1|C) * p(x_2|C) * \ldots * p(x_d|C)$$

It also assumes that the pdf $p(x_j|C)$ is a Gaussian pdf. It estimates this pdf by estimating the mean μ and the variance σ^2 of the distribution from the training examples.

For each test example $(x_1, \ldots, x_d)$, you will need to find the class maximizing

$$p(x_1|C) * p(x_2|C) * \ldots * p(x_d|C) * P(C)$$

using the estimates of the $p(x_j|C)$ and $p(C)$.

Training details: During the training phase, you will use the examples in the training file to estimate the value of $P(C)$ for each class. For each pair x_j, C

[1] The original dataset is taken from the UCIrvine data repository, and is available at `https://archive.ics.uci.edu/ml/datasets/spambase`. We have modified the dataset slightly for this assignment, so you should use the version that is posted on NYU Classes. Do not use the original version of the dataset for this assignment.

of attribute and class, you will need to estimate the mean and the Gaussian for the pdf $p(x_j|C)$.

Estimation of $P(C)$: To estimate the $P(C)$ values, just calculate the fraction of the training examples that are in class C. For example, to estimate $P(C = 1)$ calculate:

$$\frac{\text{number of examples in training file in Class 1}}{\text{total number of examples in training file}}$$

Estimation of parameters of Gaussian pdf $p(x_i|C)$: Find all examples in the training file that are in class C. Let $\mathcal{X}_C$ denote this set. Let N_C be the number of examples in this set.

To estimate the mean $\hat{\mu}$ of the Gaussian pdf $p(x_i|C)$, just calculate the average value of attribute x_i among all examples in $\mathcal{X}_C$.

To estimate the variance, use the following formula:

$$\hat{\sigma}^2 = \frac{\sum_{(x^t,r^t)\in\mathcal{X}_C}(x_t - \hat{\mu})^2}{N_C - 1}$$

where $\hat{\mu}$ is the estimate of the mean that you just calculated.

Note that $N_C - 1$ is in the denominator here, not N_C. (We'll explain why later.)

Once these values are computed, the resulting estimated pdf is

$$p(x_i|C) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\hat{\mu})^2}{2\hat{\sigma}^2}} \tag{1}$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ are the values calculated for this x_j, C.

(Make sure to calculate a new $\hat{\mu}$ and $\hat{\sigma}^2$ for each x_j, C pair.)

Testing details: For each example $(x_1, \ldots, x_9)$ in the test file, you want to determine which class maximizes

$$p(x_1|C) * p(x_2|C) * \ldots * p(x_d|C) * P(C)$$

Use ONLY attributes 1 through 9, when doing testing. Do not accidentally include the label! Compute a value for each $p(x_j|C)$ using the pdf in Equation 1, with the mean and variance you estimated for pair x_j, C during training.

Multiplying lots of small values can lead to underflow. To avoid that, you should not calculate

$$p(x_1|C) * p(x_2|C) * \ldots * p(x_9|C) * P(C)$$

directly for each class C. Instead, calculate $\log[p(x|C) * P(C)] = \log P(C) + \sum_{i=1}^{9} \log p(x_9|C)$. (Use the natural log, ln.) Then label the example with the class achieving the maximum value for this expression. If there is a tie, give the example the label 1.

Outputs

Your program should output the following values and write them to a separate text file (or the standard output) which you will NOT hand in.

- The estimated value of $P(C)$ for each class C.
- The estimates $(\hat{\mu}, \hat{\sigma}^2)$ for the Gaussians corresponding to $p(x_i|C)$, for each attribute x_i and each class C_i. (so you need to output 18 pairs $(\hat{\mu}, \widehat{\sigma^2})$).
- The predicted classes for all the test examples.
- Total number of test examples classified correctly. (You need to compare the predicted class for each test example with the given class label.)
- Total number of test examples classified incorrectly.
- The percentage error on the test examples.

Questions

Answers the following questions and put your answers in a pdf file called `proganswers.pdf`.

1. What was the estimated value of $P(C)$ for $C = 1$?
2. What was the estimated value of $P(C)$ for $C = 0$?
3. What were the estimated values for $(\hat{\mu}, \widehat{\sigma^2})$ for the Gaussian corresponding to attribute `capital_run_length_longest` and class 1 (Spam).
4. What were the estimated values for $(\hat{\mu}, \widehat{\sigma^2})$ for the Gaussian corresponding to attribute `char_freq_;` and Class 0.
5. Which classes were predicted for the first 5 examples in the test set?
6. Which classes were predicted for the last 5 examples in the test set?
7. What was the percentage error on the examples in the test file?
8. Sometimes a not-very-intelligent learning algorithm can achieve high accuracy on a particular learning task simply because the task is easy. To check for this, you can compare the performance of your algorithm to the performance of some very simple algorithms. One such algorithm just predicts the majority class (the class that is most frequent in the training set). This algorithm is sometimes called Zero-R. It can achieve high accuracy in a 2-class problem if the dataset is very imbalanced (i.e., if the fraction of examples in one class is much larger than the fraction of examples in the other). What accuracy is attained is you use Zero-R instead of Gaussian Naive Bayes?

9. Gaussian Naive Bayes is based on two assumptions: (1) the conditional independence assumption, and (2) the assumption that the pdfs for $p(x_j|C)$ are Gaussian. These assumptions are more reasonable for some datasets than for others. Do you think these assumptions are reasonable for the spam dataset you just used? Why or why not? In answering this question, you can give a common-sense argument and/or show relevant plots, graphs, or statistical information.

   (Note that Gaussian Naive Bayes can sometimes be effective even if the assumptions are not very reasonable. In order to do correct classification, it is enough to determine the correct MAP class. It is not necessary to actually compute the correct posterior probability $P(C|x)$ for each class.)

What to Hand In

Hand in a zip file which contains all the files needed to run your program, AND the file `proganswers.pdf`. If it is not obvious how to run your program, you should include a README file.

Do not include the `spamtrain.csv, spamtest.csv, and spambase1.txt` in your zip file.