Naive Bayes Classifier

36-290 – Statistical Research Methodology

Week 10 Tuesday – Fall 2021

The Model

The Naive Bayes classifier is a simple probabilistic classifier that has been around for 50+ years and is a popular baseline model for text classification, particularly spam detection. Why it is called "naive" and "Bayes" will become more clear below.

Naive Bayes is a *conditional probability model*: given a vector of predictor variable values $\mathbf{x}$, the algorithm assigns conditional probabilities for each of the response variable's K classes:

$$p(C_k|\mathbf{x}) \, .$$

The conventional decision rule is the so-called *MAP*, or *maximum a posteriori* rule: pick the class that is most probable.

But: how does one estimate $p(C_k|\mathbf{x})$?

The Derivation

The first step is to apply Bayes' rule from probability theory (hence, the "Bayes"):

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})} \propto p(C_k)p(\mathbf{x}|C_k)$$

We do not care about the value of the denominator, which is a constant in any given analysis.

The next step is to expand $p(\mathbf{x}|C_k)$:

$$\begin{aligned} p(\mathbf{x}|C_k) &= p(x_1, \ldots, x_p|C_k) \\ &= p(x_1|x_2, \ldots, x_p, C_k)p(x_2|x_3, \ldots, x_p, C_k) \cdots p(x_p|C_k) \end{aligned}$$

The third step is where the "naive" aspect of the classifier comes into play. We assume (perhaps correctly, but probably incorrectly) that the predictor variables are all mutually independent, i.e., that

$$p(x_1|x_2, \ldots, x_p, C_k)p(x_2|x_3, \ldots, x_p, C_k) \cdots p(x_p|C_k) \rightarrow p(x_1|C_k) \cdots p(x_p|C_k)$$

So in the end:

$$p(C_k|\mathbf{x}) \propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \, .$$

Further Assumptions

To utilize Naive Bayes, one needs to assign "prior probabilities" $p(C_k)$ and needs to assume conditional distributions for each class:

- Common choices for $p(C_k)$ are $1/K$ (equal probabilities for each class) and n_k/N (the number of training data in class k divided by the training set sample size).
- As for $p(x_i|C_k)$:
  - if x_i is a quantitative variable, one often assumes that $p(x_i|C_k)$ is a normal distribution, with mean and variance given by the sample mean and sample variance of the training data in class k; or
  - if x_i is a categorial variable, one often assumes that $p(x_i|C_k)$ is a binomial distribution (if there are two categories) or a multinomial distribution (if there are more than two categories), with the relative proportions of each category informing the category probability estimate.

Bottom Line

Why use Naive Bayes?

- Because of the assumption of mutual independence, the mathematics is considerably simplified and the algorithm is thus *fast*. This is especially helpful for large datasets.

Why not use Naive Bayes?

- The assumption of mutual independence would rarely hold in practice. Thus one sacrifices information about the joint distribution of predictor variables for computational speed.

$\Rightarrow$ Given its speed and ease of implementation, it never hurts to try Naive Bayes out. Do not expect it to win the misclassification error battle…but be happy if it does!

Naive Bayes: Example

We'll begin by importing data on 500 stars and 500 quasars:

```
##      col.ug           col.gr            col.ri            col.iz            mag.r         class    
##  Min.   :-4.2274   Min.   :-2.98092   Min.   :-0.40610   Min.   :-3.69967   Min.   :14.43   QSO :500  
##  1st Qu.: 0.6613   1st Qu.: 0.09591   1st Qu.: 0.02866   1st Qu.: 0.02976   1st Qu.:17.95   STAR:500  
##  Median : 1.1102   Median : 0.26471   Median : 0.12162   Median : 0.14411   Median :18.75             
##  Mean   : 1.3196   Mean   : 0.37682   Mean   : 0.21581   Mean   : 0.18544   Mean   :18.66             
##  3rd Qu.: 1.7465   3rd Qu.: 0.51801   3rd Qu.: 0.25169   3rd Qu.: 0.29248   3rd Qu.:19.47             
##  Max.   : 6.2807   Max.   : 2.68311   Max.   : 3.39274   Max.   : 4.04392   Max.   :24.82             
```

We will use the functions of the `e1071` package below, after performing a 70-30 data split. Note that there are other packages that you could utilize as well, such as `naivebayes`.

Naive Bayes: Example

```
library(e1071)

nb.out = naiveBayes(class~.,data=df.train)
nb.pred = predict(nb.out,newdata=df.test,type="class") # class is fine for 50/50 data; use raw otherwise
# e.g., nb.prob = predict(nb.out,newdata=df.test,type="raw")[,2] for Class 1 probabilities (for ROC, etc.)
table(nb.pred,df.test$class)
```

```
##
## nb.pred QSO STAR
##    QSO  136   46
##    STAR  16  102
```

```
mean(nb.pred!=df.test$class)
```

```
## [1] 0.2066667
```

Compare with...

```
log.out = glm(class~.,data=df.train,family=binomial)
log.prob = predict(log.out,newdata=df.test,type="response")
log.pred = ifelse(log.prob>0.5,"STAR","QSO")
table(log.pred,df.test$class)
```

```
##
## log.pred QSO STAR
##     QSO  132   33
##     STAR  20  115
```

```
mean(log.pred!=df.test$class)
```

```
## [1] 0.1766667
```