Bayes Estimators and Bayes Classifier

Sean Tey, Shyam Sudhakaran, Rock Gu

2018-05-08

Contents

1 Abstract

2 Bayesian Inference

2.1 Bayes Estimators

2.1.1 Point Estimation

Bayesian Estimation of Binomial Distribution's Parameter

Bayes Rule says:

$$\begin{aligned} P(\theta|y) &= \frac{P(y|\theta)P(\theta)}{P(y)} \\ &= \frac{P(y|\theta)P(\theta)}{\int P(y|\theta)P(\theta)d\theta} \quad \text{(By law of total probability)} \end{aligned}$$

$Y \sim BIN(n, \theta)$, we want to find a bayes estimator for θ, which we can get from $\hat{\theta}_{Bayes} = T = E[P(\theta|y)]$ which is the expected value of the posterior. To use the bayes rule, we have to assume a prior distribution for θ. For simplicity in the calculation, we will assume $\theta \sim Beta(\alpha, \beta)$ where α and β are known.

So now we will calculate: $P(\theta|y) = \frac{P(y|\theta)P(\theta)}{\int P(y|\theta)P(\theta)d\theta}$ by first calculating the denominator $P(y) = \int P(y|\theta)P(\theta)d\theta$

$$\begin{aligned} \int P(y|\theta)P(\theta)d\theta &= \int \binom{n}{y}\theta^y(1-\theta)^{n-y}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta \\ &= \binom{n}{y}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int \theta^y(1-\theta)^{n-y}\theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta \\ &\text{Define } \binom{n}{y}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \text{ to be } c_1. \\ &= c_1\int \theta^{\alpha+y-1}(1-\theta)^{\beta+n-y-1}d\theta \\ &\text{Recall that the CDF of the Beta Distribution is } = \frac{\int x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}dx \\ &\text{Define } B(\alpha_0,\beta_0) = \frac{\Gamma(\alpha_0)\Gamma(\beta_0)}{\Gamma(\alpha_0+\beta_0)}, \text{ where for our equation } \alpha_0 = \alpha+y-1 \text{ and } \beta_0 = \beta+n-y \\ &= c_1B(\alpha_0,\beta_0)\int \frac{\theta^{\alpha+y-1}(1-\theta)^{\beta+n-y-1}}{B(\alpha_0,\beta_0)}d\theta \text{ (Multiply equation with } \frac{B(\alpha,\beta)}{B(\alpha,\beta)} = 1) \\ &= c_1B(\alpha_0,\beta_0)\int \frac{\theta^{\alpha+y-1}(1-\theta)^{\beta+n-y-1}}{B(\alpha_0,\beta_0)}d\theta \to 1 \\ &= c_1B(\alpha_0,\beta_0) \\ &= \binom{n}{y}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \times \frac{\Gamma(\alpha+y-1)\Gamma(\beta+n-y)}{\Gamma(\alpha+\beta+n)} \\ &= P(y) \end{aligned}$$

Once we've calculated $P(y)$ we can now calculate $P(\theta|y)$.

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{\int P(y|\theta)P(\theta)d\theta}$$

2.1.2 Interval Estimation

2.1.3 Example in R

3 Gaussian Naive Bayes Classifier

3.1 How it works

3.1.1 Main idea

A Bayes classifier is an application of Bayes' Theorem: $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$ used to predict the likelihood of a class Y given the observed data X.

For example, let's say we are trying to predict the type(class) of fruit given the features $X = x_1, x_2, x_3$ representing color, shape, and taste. The classes of fruit in our example are grape, orange, and other. We observed that $x_1 = purple, x_2 = round, x_3 = sweet$. What is the probability that this fruit is of class "grape"?

We can estimate this by calculating:

$$\begin{aligned} P(Y = grape|X = x_1, x_2, x_3) &= \frac{P(X|Y = grape)P(Y = grape)}{P(X)} \\ &= \frac{P(x_1, x_2, x_3|grape)P(grape)}{P(X)} \\ P(Y = grape|X = purple, round, sweet) &= \frac{P(purple, round, sweet|grape)P(grape)}{P(purple, round, sweet)} \end{aligned}$$

3.1.2 Naive Assumption

There is a problem with the previous equation, because calculating $P(x_1, x_2, x_3|grape)$ can be difficult due to the joint conditional probability, especially if we have even more features. This is where the "naive" assumption comes in, we will **assume conditional independence** between the features x_1, x_2, x_3. Hence the name, Naive Bayes Classifier. By using this assumption and the chain rule for probability, this will give us:

$$\begin{aligned} P(Y = grape|X = purple, round, sweet) &= \frac{P(purple, round, sweet|grape)P(grape)}{P(X)} \\ &= \frac{P(purple|grape)P(round|grape)P(sweet|grape)P(grape)}{P(purple, round, sweet)} \end{aligned}$$

We can also manipulate $P(purple, round, sweet)$ such that it is easier to calculate using the **law of total probability**. We define the set of all classes to be k, in this fruit example, k = grape, orange, other.

$$P(X) = \sum_k P(X|Y = k)P(Y = k)$$

Giving us a new equation:

$$\begin{aligned} P(Y = grape|X = purple, round, sweet) &= \frac{P(purple, round, sweet|grape)P(grape)}{P(X)} \\ &= \frac{P(purple|grape)P(round|grape)P(sweet|grape)P(grape)}{\sum_k P(X|Y = k)P(Y = k)} \\ &= \frac{P(purple|grape)P(round|grape)P(sweet|grape)P(grape)}{P(X|grape)P(grape) + P(X|orange)P(orange) + P(X|other)P(other)} \end{aligned}$$

3.1.3 Empirical estimate for discrete values

The calculation will be much easier now that we no longer have the joint probability distributions. In fact we can get an empirical estimate of $P(x_1 = purple|Y = grape)$ by simply counting the proportion of occurences of the color purple within the subset data enties pre-labeled as "grape".

For example if the subset of observations X_i with label = "grape" is size 100 and we see the color feature to be purple for 80 of them, then the empirical estimate would be 80/100. Storing this information allows us to "train" the classifier to make future predictions on new observations.

3.1.4 Empirical estimate for continunous values

There is one last component to this puzzle which is how do we make empirical estimates for the conditional probability $P(x_i|Y = grape)$ if x_i is a continuous value. So far the features we have: color, shape, taste are discrete features. What if we incorporate continuous features to our model such as diameter?

A solution to this is to assume that $P(x_i|Y = grape)$ follows some distribution. In our case, we will assume that $P(x_i|Y = grape)$ follows a **gaussian distribution**. To calculate $P(x_i|Y = grape)$ we will need to estimate the mean and variance from a given feature with a given label. We can simple use sample mean and sample variance or try a bayes estimator. Note that if we had p features and k classes, we would need p×k number of means and another p×k variances.

Once we have the estimated means and variances, we will just calculate the probability density for each continuous entry in data X. Notice that we are calculating the density function, this makes the notation $P(x_i|Y = grape)$ inaccurate, but we will stick to it for simplicity. This brings up another question, will the value of $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$ still result in a probability ranging from 0 to 1? The answer is yes, because looking at the previously expanded equation carefully:

$$\begin{aligned} P(Y = grape|X = purple, round, sweet) &= \frac{P(X|Y = grape)P(Y = grape)}{\sum_k P(X|Y = k)P(Y = k)} \\ &= \frac{P(X|grape)P(grape)}{P(X|grape)P(grape) + P(X|orange)P(orange) + P(X|other)P(other)} \end{aligned}$$

Notice that the numerator is part of the denominator. This essentially normalizes the equation to [0,1] because at most the highest possible value for $P(Y|X)$ is 1, which is the case where $P(X|grape)P(grape) \neq 0$ and $P(X|orange)P(orange) = P(X|other)P(other) = 0$.

3.2 Application Example

3.2.1 Description

Here we will attempt to use the gaussian naive bayes classifier described previously to predict the movement of the Standard & Poor's 500 stock market index. The possible prediction for classes are Y = k where k = 0,1.

Y = 1, means the S&P500 has positively increased since the previous day.
Y = 0, means the S&P500 has decreased or has not changed since the previous day.

We will use stock market data/metrics such as price and volume as features or predictors. We define the total number of features to be p. The total number of observation is n where each observation is one trading day.

The data matrix X will be a n×p matrix and response(label) vector Y will be a n×1 vector which looks like:

$$X_{ij} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{np} \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

where i = 1,...,n trading days and j= 1...p stock metrics (features), $y_n \in 0, 1$

$$\begin{array}{l} \\ \text{TradingDay}_1 \\ \text{TradingDay}_2 \\ \\ \text{TradingDay}_\text{n} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Metric}_1 & \text{Metric}_2 & \dots & \dots & \text{Metric}_\text{p} \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{np} \end{bmatrix} \end{array}$$

$$\left[\begin{array}{c|cc} \text{metric 1} & \textit{metric}2 & \textit{metric}3 \\ x11 & x12 & x13 \end{array}\right]$$

$$\begin{array}{l} \text{tt} \\ \text{tt} \end{array} \begin{bmatrix} \begin{array}{cc} \text{metric 1} & \text{metric 2} \end{array} \\ \begin{array}{cc} x11 & x12 \\ x11 & x12 \end{array} \end{bmatrix}$$

3.2.2 R Code

All of the code can be found at https://github.com/seantey/gaussian_naive_bayes_for_sp500.

```r
# bayes_estimator flag toggles whether to use sample mean vs bayes estimator
gaussian_nb <- function(X,y,bayes_estimator=FALSE){

  gnb_check_input(X,y) # Predefined function which checks input validity

  # sum(y) gives total days which we observe increase in S&P500
  # length(y) gives total observations (i.e. days)
  prior_Y_one <- sum(y) / length(y)
  prior_Y_zero <- 1 - prior_Y_one

  # Create subset data frames where Y = 0 or Y = 1
  X_given_one_subset <- X[as.logical(y),]
  X_given_zero_subset <- X[as.logical(1-y),]

  if (bayes_estimator == FALSE){
    mew_one = sapply(X_given_one_subset,mean)
    sigma_one = sapply(X_given_one_subset,sd)

    mew_zero = sapply(X_given_zero_subset,mean)
    sigma_zero = sapply(X_given_zero_subset,sd)

  } else {
    # Some bayes estimator function output:
    mew_one = sapply(X_given_one_subset,mean) # replace mean function with something else!
    sigma_one = sapply(X_given_one_subset,sd) # replace sd function with something else!

    mew_zero = sapply(X_given_zero_subset,mean) # replace mean function with something else!
    sigma_zero = sapply(X_given_zero_subset,sd) # replace sd function with something else!

  }

  # Initialize a data frame of size n x p to hold P(Xij|Y=k) for each entry of matrix X
  X_given_Y_one <- data.frame(matrix(nrow = nrow(X),ncol = ncol(X))) # Y = 1
  X_given_Y_zero <- data.frame(matrix(nrow = nrow(X),ncol = ncol(X))) # Y = 0

  # For every individual entry Xij, calculate the prob(Xij|Y = 1),
  # recall that we are fitting a gaussian distribution for this

  # Use appropriate mew & sigma corresponding to column p and class Y = k
  for (i in 1:nrow(X)){
    row_data = X[i,]

    for (p in 1:ncol(X)){
      prob_XY <- dnorm(row_data[[p]],mew_one[p],sigma_one[p])
      X_given_Y_one[i,p] <- prob_XY
    }
  }

  # For every individual entry Xij, calculate the prob(Xij|Y = 0)
  for (i in 1:nrow(X)){
    row_data <- X[i,]

    for (p in 1:ncol(X)){
      prob_XY <- dnorm(row_data[[p]],mew_zero[p],sigma_zero[p])
      X_given_Y_zero[i,p] <- prob_XY

    }
  }
```

```
59
60    # Take product of entire row from rows 1...n
61
62    # Numerator = P(Xi1|Y=1)...P(Xip|Y=1)P(Y=1) for i = 1...n
63    row_product_ones <- apply(X_given_Y_one,1,prod)
64    numerator_ones <- row_product_ones*prior_Y_one
65
66    row_product_zeros <- apply(X_given_Y_zero,1,prod)
67    numerator_zeros <- row_product_zeros*prior_Y_zero
68
69    # Add numerator for Y=1 and Y=0 to get denominator.
70    # Denominator = P(Xi1|Y=1)...P(Xip|Y=1)P(Y=1) + P(Xi1|Y=0)...P(Xip|Y=1)P(Y=0)
71
72    # Calculate posterior probability P(Y|X)
73    posteriors_ones <- numerator_ones / (numerator_ones + numerator_zeros)
74    posteriors_zeros <- numerator_zeros / (numerator_ones + numerator_zeros)
75
76    # Predictions
77    # If P(Y=1|Xij) > P(Y=0|Xij), i.e. likelihood of class 1 is greater,
78    # then we will label the observation row i as class Y = 1
79    predictions <- ifelse(posteriors_ones>posteriors_zeros,1,0)
80
81    # Generate list of outputs: (1) Prior Probabilities of Y, (2) Class conditional
          probabilities P(Xij|Y=k), (3) Posterior probabilities, (4) Predicted Labels for
          each data row, (5) accuracy
82
83    output_priors <- c(prior_Y_zero,prior_Y_one)
84
85    class_conditional_DF_list <- list(X_given_Y_zero,X_given_Y_one)
86
87    posteriors_DF <- data.frame(posteriors_zeros,posteriors_ones)
88
89    accuracy <- sum(ifelse(predictions==y,1,0))/nrow(X)
90
91    # Generate a named list for easy access to elements.
92    # E.g. output_list$priors will access prior probabilities etc
93    output_list <- list(priors = output_priors, class_conditional_DF_list = class_
          conditional_DF_list,
94                        posteriors_DF = posteriors_DF, predictions = predictions,
95                        accuracy = accuracy)
96
97    return(output_list)
98
99  }
```

Function to check input validity. Ensures that input X is a data frame and y is a binary vectory of 1's and 0's. The function also ensures the rows(length) of X and Y are equal. E.g. If X is n×p, then y must be a n×1 vector.

```r
gnb_check_input <- function(X,y){

  if(!is.data.frame(X)) {
    stop("Parameter X must be a data frame, please provide the data frame whose columns
         are numeric variables and rows are observations")
  }

  # Check if length of vector y matches number of rows in matrix X
  if(!(nrow(X)==length(y))) stop("Length of response vector y must be equal to number
      of rows in X")

  # Helper function to verify response vector input.
  binary_vector_check <- function(input_vector){
    if(!is.integer(input_vector)) stop("Please provide an integer vector")
    is_binary <- all(input_vector <= 1) && all(input_vector >= 0)
    if(!is_binary) stop("Please provide an integer vector of 1's and 0's")
    return(TRUE)
  }

  # Check if y is made of 1's and 0's
  # Where 1 = increase in price since yesterday, 0 = decrease or no increase
  binary_vector_check(y)
}
```

3.2.3 Findings

Results from MLE estimators

Results from Bayes estimators

Conclusion