

Chapter 03: Toward NaiveBayes: Multiclass Classifier

Multivariate (MV) Probability

In a univariate scenario where Y and X are some vector, joint and conditional probabilities may be written as

$P(x,y)=P(x|y)P(y)/P(x)$ where $P(x,y)$, $P(x|y)$ and $p(y)$ are the Joint, conditional and the marginal probabilities.

Note that a univariate solution may not be applicable to model probability for a classification task most of the time because there is always more than one predictor.

Mostly, we want to determine $P(\text{class}|\text{data})$ and data is almost always multivariate (MV) not univariate.

$P(\text{class}|\text{data})P(\text{data})=P(\text{class},\text{data})$ and $P(\text{data},\text{class})=P(\text{data}|\text{class})P(\text{class})$

Since $P(c,d)=P(d,c)$, $P(\text{class}|\text{data})P(\text{data})=P(\text{data}|\text{class})P(\text{class})$

or $P(\text{class}|\text{data})= P(\text{data}|\text{class})P(\text{class})/P(\text{data})$

Now for a Bayesian Classifier, given a new observation, we compute the class probabilities conditioned on data, we assign the class with the highest probability. Thus, exact value of the posterior $P(\text{class}|\text{data})$ is not necessary, all we need is the ration. We can avoid the cost of computing $P(\text{data})$ as it is the same for all the classes.

$$P(\text{class}|\text{data})= P(\text{data}|\text{class})P(\text{class})$$

When data is conditionally independent, and there are n predictors, we can write the above as ,

$$P(\text{data}|\text{class})=P(x_1|\text{class})P(x_2|\text{class})..P(x_n|\text{class})$$

NaïveBayes makes this assumption that the predictors are conditionally independent.

Let us say there are two classes c_1 and c_2 and three predictors x_1, x_2, x_3 , then

$P(c_1|x_1, x_2, x_3)=P(x_1|c_1)P(x_2|c_1)P(x_3|c_1)P(c_1)$ for class C_1

$P(c_2|x_1, x_2, x_3)=P(x_1|c_2)P(x_2|c_2)P(x_3|c_2)P(c_2)$ for class C_2

We can compute the numerator and assign the class with the higher value. $P(c_1)$ and $P(x|c_2)$ are easily calculated.

```
> catheart<-heart[,c(2,3,6,7,9,11,12,13,14)]
> head(catheart)
  sex cp fbs restecg exang slope ca thal target
1  1  3  1      0      0      0  0   1      1
2  1  2  0      1      0      0  0   2      1
3  0  1  0      0      0      2  0   2      1
4  1  1  0      1      0      2  0   2      1
5  0  0  0      1      1      2  0   2      1
6  1  0  0      1      0      1  0   1      1
```

```
heart<-read.csv('C:/Users/rkannan/rk/data/heart.csv',head=T,sep=',',
stringsAsFactors=F)
catheart<-heart[,c(2,3,6,7,9,11,12,13,14)]
```

$P(c_1)$ =proportion of c_1 =Number of occurrences of C_1 /Total Occurrences in the dataset

```
TBL<-table(catheart$target)
```

```
CL<-names(TBL)
print(paste("P(c=",CL[1],")=",TBL[[1]]/sum(TBL),sep=""))
print(paste("P(c=",CL[2],")=",TBL[[2]]/sum(TBL),sep=""))
```

```
> TBL<-table(catheart$target)
> CL<-names(TBL)
> print(paste("P(c=",CL[1],")=",TBL[[1]]/sum(TBL),sep=""))
[1] "P(c=0)=0.455445544554455"
> print(paste("P(c=",CL[2],")=",TBL[[2]]/sum(TBL),sep=""))
[1] "P(c=1)=0.544554455445545"
```

These are the prior class probabilities.

Let us write a function to calculate,

```
nb_likelihood<-function(df,label,class,feature,val)
{nrow(df[df[[feature]]==val&df[[label]]==class,])/nrow(df[df[[label]]==class,)]}
```

Here, `df[df[[feature]]==val&df[[label]]==class,]` subsets the `df` by the rows that satisfy the specified condition. `nrow()` returns the number of rows. So `nb_likelihood` function returns the proportion of the records for a specified class.

```
tstidx<-sample(1:nrow(catheart),0.3*nrow(catheart),replace=F)
trcatheart<-heart[-tstidx,]
nb_likelihood(trcatheart,'target',0,'cp',2)
[1] 0.1318681
> nb_likelihood(trcatheart,'target',1,'cp',2)
[1] 0.4380165
If we were to classify using this single feature cp, when cp==2, that instance belongs to class 1,
because the probability is higher
> nb_likelihood(trcatheart,'target',0,'cp',0)
[1] 0.7582418
> nb_likelihood(trcatheart,'target',1,'cp',0)
[1] 0.214876
> nb_likelihood(trcatheart,'target',0,'cp',3)
[1] 0.05494505
> nb_likelihood(trcatheart,'target',1,'cp',3)
[1] 0.107438
> nb_likelihood(trcatheart,'target',0,'cp',1)
[1] 0.05494505
> nb_likelihood(trcatheart,'target',1,'cp',1)
[1] 0.2396694
nrow(catheart[catheart$sex==1&catheart$target==0,])/nrow(catheart[catheart$target==0,])
[1] 0.826087
nrow(catheart[catheart$sex==0&catheart$target==0,])/nrow(catheart[catheart$target==0,])
[1] 0.173913
nrow(catheart[catheart$sex==1&catheart$target==1,])/nrow(catheart[catheart$target==1,])
[1] 0.5636364
nrow(catheart[catheart$sex==0&catheart$target==1,])/nrow(catheart[catheart$target==1,])
[1] 0.4363636
```

```

> nrow(catheart[catheart$sex==1&catheart$target==0,])/nrow(catheart[catheart$target==0,])
[1] 0.826087
> nrow(catheart[catheart$sex==0&catheart$target==0,])/nrow(catheart[catheart$target==0,])
[1] 0.173913
> nrow(catheart[catheart$sex==1&catheart$target==1,])/nrow(catheart[catheart$target==1,])
[1] 0.5636364
> nrow(catheart[catheart$sex==0&catheart$target==1,])/nrow(catheart[catheart$target==1,])
[1] 0.4363636

nrow(trcatheart[trcatheart$sex==1&trcatheart$target==0,])/nrow(trcatheart[trcatheart$target==0,])
nrow(trcatheart[trcatheart$sex==0&trcatheart$target==0,])/nrow(trcatheart[trcatheart$target==0,])
nrow(trcatheart[trcatheart$sex==1&trcatheart$target==1,])/nrow(trcatheart[trcatheart$target==1,])
nrow(trcatheart[trcatheart$sex==0&trcatheart$target==1,])/nrow(trcatheart[trcatheart$target==1,])
  table(catheart[catheart$sex&catheart$target==0,c('sex','target')])/
  dim(catheart[catheart$target==0,c('sex','target')])[1]
library(e1071)
nbmodel<-naiveBayes(target~.,data=catheart)
nb.predicted<-predict(nbmodel,catheart[, -which(names(catheart)=='target')],type='raw')
nb.pred<-unlist(lapply(apply(nb.predicted,1,which.max),
  FUN=function(x)names(as.data.frame(nb.predicted))[[x]]))
> table(catheart$target,nb.pred)
.
.
.
> nb.pred<-unlist(lapply(apply(nb.predicted,1,which.max),FUN=function(x)names(as.data.frame(nb.predicted))[[x]]))
> table(catheart$target,nb.pred)
  nb.pred
    0    1
0 113   25
1   22 143

cfm<-caret::confusionMatrix(table(catheart$target,nb.pred))
cfm
> cfm<-caret::confusionMatrix(table(catheart$target,nb.pred))
> cfm
Confusion Matrix and Statistics

  nb.pred
    0    1
0 113   25
1   22 143

Accuracy : 0.8449
 95% CI : (0.7991, 0.8837)
No Information Rate : 0.5545
P-Value [Acc > NIR] : <2e-16

Kappa : 0.6867

McNemar's Test P-Value : 0.7705

Sensitivity : 0.8370
Specificity : 0.8512
Pos Pred Value : 0.8188
Neg Pred Value : 0.8667
Prevalence : 0.4455
Detection Rate : 0.3729
Detection Prevalence : 0.4554
Balanced Accuracy : 0.8441

'Positive' Class : 0
> |

```

Now let us split the data into training/test sets and

```

set.seed(43)
trdidx<-sample(1:nrow(heart.df),0.7*nrow(heart.df),replace=F)
trheart.df<-heart.df[trdidx,]
tsheart.df<-heart.df[-trdidx,]
nbtr.model<-naiveBayes(target~.,data=trheart.df)
nbtr.pred<-predict(nbtr.model,tsheart.df[, -c(6)],type='raw')
nbtr.class<-unlist(apply(round(nbtr.pred),1,which.max))-1
nbtr.tbl<-table(tsheart.df[[6]], nbtr.class)
stst.cfm<-caret::confusionMatrix(nbtr.tbl)
tr.cfm
nbtr.pred<-predict(nbtr.model,tscatheart[, -c(9)],type='raw')

```

```

nbtr.class<-unlist(apply(round(nbtr.pred),1,which.max))-1
nbtr.tbl<-table(tstcatheart[[9]], nbtr.class)
tst.cfm<-caret::confusionMatrix(nbtr.tbl)
tst.cfm

```

```

> tst.cfm
Confusion Matrix and Statistics

```

```

      nbtr.class
      0      1
0 32 15
1  8 36

      Accuracy : 0.7473
      95% CI   : (0.6453, 0.8325)
No Information Rate : 0.5604
P-Value [Acc > NIR] : 0.0001746

      Kappa : 0.4965

McNemar's Test P-Value : 0.2109029

      Sensitivity : 0.8000
      Specificity : 0.7059
      Pos Pred Value : 0.6809
      Neg Pred Value : 0.8182
      Prevalence : 0.4396
      Detection Rate : 0.3516
      Detection Prevalence : 0.5165
      Balanced Accuracy : 0.7529

      'Positive' Class : 0

```

```

> tst.cfm<-caret::confusionMatrix(nbtr.tbl)
> tst.cfm
Confusion Matrix and Statistics

```

```

      nbtr.class
      0      1
0 38  9
1  5 39

      Accuracy : 0.8462
      95% CI   : (0.7554, 0.9133)
No Information Rate : 0.5275
P-Value [Acc > NIR] : 1.496e-10

      Kappa : 0.6929

McNemar's Test P-Value : 0.4227

      Sensitivity : 0.8837
      Specificity : 0.8125
      Pos Pred Value : 0.8085
      Neg Pred Value : 0.8864
      Prevalence : 0.4725
      Detection Rate : 0.4176
      Detection Prevalence : 0.5165
      Balanced Accuracy : 0.8481

      'Positive' Class : 0

```

```

catheart<-heart[,c(2,3,6,7,9,11,12,13,14)]
head(catheart)
table(catheart$target)
sum(table(catheart$target))
set.seed(43)
trdidx<-sample(1:nrow(catheart),
0.7*nrow(catheart),replace=F)
trcatheart<-catheart[trdidx,]
tstcatheart<-catheart[-trdidx,]
nbtr.model<-naiveBayes(target~.,data=trcatheart)
nblearning.pred<-predict(nbtr.model,trcatheart[,c(9)],type="raw")
nblearning.class<-unlist(apply(round(
nblearning.pred),1,which.max))-1
nblearning.tbl<-table(trcatheart[,c(9)],nblearning.class)
tr.cfm<-caret::confusionMatrix(nblearning.tbl)

```

In R, iterative computations can be vectorized as follows

-----formatt correcting--- vectorized R solution-----

```

do.call("rbind",
lapply(c("sex", "cp", "fbs", "restecg", "exang", "slope", "ca", "thal"),
FUN=function(d,df=catheart,cl='target') {
classvalues=unique(df[[cl]])

```

```

do.call('rbind',
lapply(classvalues,FUN=function(clv,feature=d,dataset=df,target=cl) {
featurevalues=unique(dataset[[feature]])
do.call('rbind',
lapply(featurevalues,FUN=function(fv,cvalue=clv,f=feature,dset=dataset,cl=target) {
c(clvar=cl,clval=clv,predictor=f,predval=fv,class_cond_prob=
nb_likelihoood(df=dset,label=cl,class=cvalue,feature=f,val=fv))
})
})#do.call
})
})#dp.call
})#lapply
})#do.call

```

[TODO] We still have to validate our implementation against standard implementation by evaluating the class probabilities for a few test data observations.