Chapter 4 Exercises: Classification

Statistical Learning with R

Jon Geiger

February 2, 2022

Conceptual Exercise 4

It was stated in the text that classifying an observation to the class for which (4.17) is largest is equivalent to classifying an observation to the class for which (4.18) is largest. Prove that this is the case. In other words, under the assumption that the observations in the kth class are drawn from a $N(\mu_k, \sigma^2)$ distribution, the Bayes classifier assigns an observation to the class for which the discriminant function is maximized.

Solution

Equation 4.17 gives us the posterior probability that a new observation will belong to the kth class, given by:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{\ell=1}^{K} \pi_\ell \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_\ell)^2\right)}.$$

We can notice that since we are trying to choose the value of k which maximizes $p_k(x)$, and that the denominator of this function is constant across ks, maximizing this posterior probability on k is equivalent to maximizing its numerator on k. Starting with this, we can derive the discriminant term, calling the

numerator something different:

$$
\begin{aligned}
p_k(x) &= \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{\ell=1}^{K} \pi_\ell \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_\ell)^2\right)} \\
\text{num}_k(x) &= \pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right) \\
\log(\text{num}_k(x)) &= \log\left(\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \log\left(\pi_k\right) + \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + \log\left(\exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)\right) \\
&= \log(\pi_k) - \log\left(\sqrt{2\pi}\sigma\right) - \frac{1}{2\sigma^2}(x-\mu_k)^2
\end{aligned}
$$

Since the middle term is the same across all classes k, we can ignore it:

$$
\begin{aligned}
&= \log(\pi_k) - \frac{1}{2\sigma^2}(x-\mu_k)^2 \\
&= \log(\pi_k) - \frac{1}{2\sigma^2}(x^2 - 2x\cdot\mu_k + \mu_k^2) \\
&= \log(\pi_k) - \frac{1}{2\sigma^2}(x^2) + \frac{1}{2\sigma^2}\cdot 2x\mu_k + \frac{1}{2\sigma^2}\cdot\mu_k^2)
\end{aligned}
$$

Ignoring the second term by the same logic:

$$
\begin{aligned}
&= \log(\pi_k) + \frac{\mu_k}{\sigma^2}\cdot x + \frac{\mu_k^2}{2\sigma^2} \\
&= x\cdot\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \\
&= \delta_k(x)
\end{aligned}
$$

Applied Exercise 16

Using the `Boston` data set, fit classification models in order to predict whether a given census tract has a crime rate above or below the median. Explore logistic regression, LDA, naive Bayes, and KNN models using various subsets of the predictors. Describe your findings.

Hint: You will have to create the response variable yourself, using the variables that are contained in the `Boston` data set.

Solution

Let's just take a quick look at the data first.

```
glimpse(Boston)
```

```
## Rows: 506
## Columns: 13
## $ crim    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829,~
## $ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## $ chas    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524,~
## $ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631,~
## $ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
```

```
## $ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ rad     <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,~
## $ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
## $ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
```

We can see that the first column of this data set is `crim`, so we can construct another column which indicates whether the crime rate is above or below the median crime rate:

```
median(Boston$crim)
```

```
## [1] 0.25651
```

```
bos <- Boston %>%
    mutate("crimeful" = crim > median(crim),
           .after = crim)
dim(bos)
```

```
## [1] 506  14
```

Now that we've created our variable, let's create a few different subsets of the data for the purposes of testing our models. We'll create these as logical columns. For the first subset, we'll use the first 400 rows as a training set with the last 106 as a test set. For the second, we'll use the last 400 rows of the data as training with the first as a test set, and for the third subset we'll randomly assign 400 of the observations to be training examples with the rest being a part of the test set.

```
bos <- bos %>%
    mutate(
        train1 = c(rep(TRUE, 400), rep(FALSE, 106)),
        train2 = c(rep(FALSE, 106), rep(TRUE, 400)),
        train3 = sample(train1),
        .before = everything()
    )
```

We can now look again at our data, which has three train/test subsets attached to it:

```
glimpse(bos[,1:3])
```

```
## Rows: 506
## Columns: 3
## $ train1 <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRU~
## $ train2 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ~
## $ train3 <lgl> FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE, TRUE,~
```

As we can see, our train-test split worked out nicely. We can now go on to using logistic regression for classifying/predicting whether the crime rate is above or below the median.

Logistic Regression

I want to take two different approaches to model selection, the first of which will be somewhat experimental and makes intuitive sense to me. The first approach is this:

1. Fit a logistic model to the whole data set with all features, and eliminate the features which were not significant
2. Take three subsets of the data as training sets, and compare variable significance in the three different training subsets. Eliminate the features which are insignificant in two or more of the subsets.
3. Fit a model with the remaining features on the random subset (subset 3) and eliminate any insignificant features.
4. Repeat step 3 until all remaining features are significant.

The second approach will only utilize a random subset, which is the `train3` column in the data.

We'll start by fitting a logistic regression model to the whole dataset to see which variables we should include in our regression model:

```
logistic_fit_initial <- glm(
    crimeful ~
        zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + lstat + medv,
    family = binomial, data = bos
)
summary(logistic_fit_initial)
```

```
##
## Call:
## glm(formula = crimeful ~ zn + indus + chas + nox + rm + age +
##     dis + rad + tax + ptratio + lstat + medv, family = binomial,
##     data = bos)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0082  -0.1688  -0.0004   0.0027   3.4324
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -38.832417   6.086280  -6.380 1.77e-10 ***
## zn           -0.086228   0.034090  -2.529   0.0114 *
## indus        -0.052438   0.042817  -1.225   0.2207
## chas          0.619914   0.722150   0.858   0.3907
## nox          47.913820   7.344213   6.524 6.84e-11 ***
## rm           -0.271941   0.676239  -0.402   0.6876
## age           0.021474   0.012105   1.774   0.0761 .
## dis           0.669991   0.214618   3.122   0.0018 **
## rad           0.669240   0.151742   4.410 1.03e-05 ***
## tax          -0.006165   0.002622  -2.351   0.0187 *
## ptratio       0.326433   0.116296   2.807   0.0050 **
## lstat         0.053537   0.047105   1.137   0.2557
## medv          0.147987   0.064347   2.300   0.0215 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.46  on 505  degrees of freedom
## Residual deviance: 218.75  on 493  degrees of freedom
## AIC: 244.75
##
```

```
## Number of Fisher Scoring iterations: 9
```

As we can see, there are a few variables which are very statistically significant, so we can use reverse selection to only focus on those variables which are significant for the model, taking a threshold of $\alpha = 0.05$. Additionally, I would like to look at the model performance based on each of the three training sets, so we'll look at the significance levels of each model based on how the data was subsetted:

```
logistic_fit_1 <- glm(
    crimeful ~ zn + nox + dis + rad + tax + ptratio + medv,
    family = binomial, data = bos, subset = bos$train1
)
logistic_fit_2 <- glm(
    crimeful ~ zn + nox + dis + rad + tax + ptratio + medv,
    family = binomial, data = bos, subset = bos$train2
)
logistic_fit_3 <- glm(
    crimeful ~ zn + nox + dis + rad + tax + ptratio + medv,
    family = binomial, data = bos, subset = bos$train3
)
```

Now that we've created the three different models with the three different subsetting techniques, let's see how our model variables hold up. Below will be a significance table which shows the coefficients for the three different models, as well as that same table in logical form, using the same cutoff of $\alpha = 0.05$.

```
signif_table <- rbind(
    summary(logistic_fit_1)$coefficients[, 4],
    summary(logistic_fit_2)$coefficients[, 4],
    summary(logistic_fit_3)$coefficients[, 4]
)
signif_table
```

```
##       (Intercept)         zn          nox        dis          rad        tax
## [1,] 1.742754e-10 0.04335784 9.085971e-12 0.01410174 6.805118e-05 0.57731492
## [2,] 7.437243e-07 0.07518172 3.762286e-08 0.10576705 1.022221e-05 0.06850258
## [3,] 2.252106e-08 0.04653104 9.625761e-10 0.03573056 2.658904e-06 0.01975819
##          ptratio        medv
## [1,] 0.001722488 0.004777056
## [2,] 0.897406027 0.019541069
## [3,] 0.064642862 0.032054133
```

```
signif_table < 0.05
```

```
##      (Intercept)    zn  nox   dis  rad   tax ptratio medv
## [1,]        TRUE  TRUE TRUE  TRUE TRUE FALSE    TRUE TRUE
## [2,]        TRUE FALSE TRUE FALSE TRUE FALSE   FALSE TRUE
## [3,]        TRUE  TRUE TRUE  TRUE TRUE  TRUE   FALSE TRUE
```

We will modify the model again using only the variables for which at least two of the models show significance, using the random subsetting technique.

```
logistic_fit_first <- glm(
    crimeful ~ zn + nox + rad + ptratio + medv,
    family = binomial, data = bos, subset = bos$train3
)
summary(logistic_fit_first)
```

```
##
## Call:
## glm(formula = crimeful ~ zn + nox + rad + ptratio + medv, family = binomial,
##     data = bos, subset = bos$train3)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.05857  -0.27912  -0.01658   0.00277   2.87909
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -23.47920    4.37663  -5.365 8.11e-08 ***
## zn           -0.03420    0.02255  -1.517   0.1294
## nox          28.10702    4.31089   6.520 7.03e-11 ***
## rad           0.59045    0.12765   4.625 3.74e-06 ***
## ptratio       0.19937    0.11225   1.776   0.0757 .
## medv          0.07173    0.03214   2.231   0.0257 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.27  on 399  degrees of freedom
## Residual deviance: 190.78  on 394  degrees of freedom
## AIC: 202.78
##
## Number of Fisher Scoring iterations: 8
```

We can now notice that suddenly, the `ptratio` variable doesn't meet the significance criterion, so we can remove it and give another summary:

```
logistic_fit_first <- glm(
    crimeful ~ zn + nox + rad + medv,
    family = binomial, data = bos, subset = bos$train3
)
summary(logistic_fit_first)
```

```
##
## Call:
## glm(formula = crimeful ~ zn + nox + rad + medv, family = binomial,
##     data = bos, subset = bos$train3)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.93515  -0.32277  -0.02098   0.00500   2.82607
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -17.45954    2.39861  -7.279 3.36e-13 ***
## zn           -0.04363    0.02295  -1.901   0.0573 .
## nox          25.53737    3.85342   6.627 3.42e-11 ***
## rad           0.54497    0.12419   4.388 1.14e-05 ***
## medv          0.04138    0.02691   1.538   0.1241
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.27  on 399  degrees of freedom
## Residual deviance: 194.01  on 395  degrees of freedom
## AIC: 204.01
##
## Number of Fisher Scoring iterations: 8
```

We now see the same thing with the `medv` variable, so we eliminate it as well:

```
logistic_fit_first <- glm(
    crimeful ~ zn + nox + rad,
    family = binomial, data = bos, subset = bos$train3
)
summary(logistic_fit_first)
```

```
##
## Call:
## glm(formula = crimeful ~ zn + nox + rad, family = binomial, data = bos,
##     subset = bos$train3)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.93512  -0.28525  -0.02450   0.00309   2.77351
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -16.58905    2.27644  -7.287 3.16e-13 ***
## zn           -0.03560    0.02089  -1.704   0.0884 .
## nox          25.24335    3.79207   6.657 2.80e-11 ***
## rad           0.58785    0.12199   4.819 1.45e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.27  on 399  degrees of freedom
## Residual deviance: 196.45  on 396  degrees of freedom
## AIC: 204.45
##
## Number of Fisher Scoring iterations: 8
```

And now we are left with a logistic regression model to predict whether or not an area is `crimeful` based on `zn`, `nox`, and `rad`.

Let's now use traditional reverse selection with the random training subset to see how different of a model we end up with.

```
kept_coefs <- function(x){
    summary(x)$coefficients[,4] < 0.05
}

glm(
    crimeful ~
        zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + lstat + medv,
    family = binomial, data = bos, subset = bos$train3
) %>% kept_coefs()
```

```
## (Intercept)          zn       indus        chas         nox          rm
##        TRUE        TRUE       FALSE       FALSE        TRUE       FALSE
##         age         dis         rad         tax     ptratio       lstat
##       FALSE        TRUE        TRUE       FALSE        TRUE       FALSE
##        medv
##       FALSE
```

```
glm(
    crimeful ~ zn + indus + nox + rad + tax + medv,
    family = binomial, data = bos, subset = bos$train3
) %>% kept_coefs()
```

```
## (Intercept)          zn       indus         nox         rad         tax
##        TRUE       FALSE       FALSE        TRUE        TRUE        TRUE
##        medv
##       FALSE
```

```
glm(
    crimeful ~ zn + indus + nox + rad + tax,
    family = binomial, data = bos, subset = bos$train3
) %>% kept_coefs()
```

```
## (Intercept)          zn       indus         nox         rad         tax
##        TRUE       FALSE       FALSE        TRUE        TRUE        TRUE
```

```
glm(
    crimeful ~ zn + nox + rad + tax,
    family = binomial, data = bos, subset = bos$train3
) %>% kept_coefs()
```

```
## (Intercept)          zn         nox         rad         tax
##        TRUE       FALSE        TRUE        TRUE        TRUE
```

```
logistic_fit_second <- glm(
    crimeful ~ zn + nox + rad + tax,
    family = binomial, data = bos, subset = bos$train3
)
```

We now have a model for the `crimeful` variable which depends on `zn`, `nox`, `rad`, and `tax`. Notice that in our previous model, we did not include tax, because it was not significant in the first two subsets, but it was significant in the third, random subset. We'll now compare the models using the first and second methods using the test set using a probability threshold of 0.5 for the logistic model.

```
logistic_preds_first  <- predict(
    logistic_fit_first,
    newdata = bos[!bos$train3, ], #make prediction on test data
    type = "response"
) > 0.5
logistic_preds_second <- predict(
    logistic_fit_second,
    newdata = bos[!bos$train3, ],
    type = "response"
) > 0.5
```

Taking a look at the confusion matrices for the first and second models, we see:

```
table(logistic_preds_first, bos$crimeful[!bos$train3])
```

```
##
## logistic_preds_first FALSE TRUE
##                FALSE    41   12
##                TRUE      7   46
```

```
table(logistic_preds_second, bos$crimeful[!bos$train3])
```

```
##
## logistic_preds_second FALSE TRUE
##                 FALSE    42   12
##                 TRUE      6   46
```

The model accuracies are then given by:

```
cat(
    "Model 1 Accuracy: ",
    round(100*mean(logistic_preds_first == bos$crimeful[!bos$train3]), 2),
    "% (crimeful ~ zn + nox + rad)", "\n",
    "Model 2 Accuracy: ",
    round(100*mean(logistic_preds_second == bos$crimeful[!bos$train3]), 2),
    "% (crimeful ~ zn + nox + rad + tax)", sep = ""
)
```

```
## Model 1 Accuracy: 82.08% (crimeful ~ zn + nox + rad)
## Model 2 Accuracy: 83.02% (crimeful ~ zn + nox + rad + tax)
```

After having run this a few times, the model accuracies have fluctuated between 75% and 90%, with both accuracies being about equal. Because of this, for other modeling techniques we will use the first model which excludes the `tax` variable for the sake of simplicity.

LDA

We will now use Linear Discriminant Analysis to fit the data using the formula from above. We will assess its accuracy as compared to logistic regression, keeping in mind that with a small test set randomly sampled (just over 100 points), the accuracy will fluctuate and may not represent the actual accuracy given a larger data set.

```
lda_fit <- lda(
    crimeful ~ zn + nox + rad,
    data = bos, subset = bos$train3
)
lda_preds <- predict(
    object = lda_fit,
    newdata = bos[!bos$train3, ]
)
lda_preds <- lda_preds$class
table(lda_preds, bos$crimeful[!bos$train3]) # Confusion Matrix
```

```
##
## lda_preds FALSE TRUE
##     FALSE    46   16
##     TRUE      2   42
```

```
cat("LDA Model Accuracy: ",
    round(100*mean(lda_preds == bos$crimeful[!bos$train3]), 2), "%", sep = ""
)
```

```
## LDA Model Accuracy: 83.02%
```

We can see that the LDA model performs practically identically to the Logistic Regression model.

QDA

Now we do the same thing as with LDA, but replace all the L's with Q's:

```
qda_fit <- qda(
    crimeful ~ zn + nox + rad,
    data = bos, subset = bos$train3
)
qda_preds <- predict(
    object = qda_fit,
    newdata = bos[!bos$train3, ]
)
qda_preds <- qda_preds$class
table(qda_preds, bos$crimeful[!bos$train3]) # Confusion Matrix
```

```
##
## qda_preds FALSE TRUE
##     FALSE    45   12
##     TRUE      3   46
```

```
cat("QDA Model Accuracy: ",
    round(100*mean(qda_preds == bos$crimeful[!bos$train3]), 2), "%", sep = ""
)
```

```
## QDA Model Accuracy: 85.85%
```

Quadratic Discriminant Analysis proves to be *slightly* more accurate than Logistic Regression or LDA, taken of course with a grain of salt knowing that accuracies will fluctuate based on the random sampling used for the training/test data.

Naive Bayes

Same thing as LDA and QDA, we can use the same process with the same syntax to calculate the naive Bayes classification on this set along with its accuracy:

```
nb_fit <- naiveBayes(
    crimeful ~ zn + nox + rad,
    data = bos, subset = bos$train3
)
nb_preds <- predict(
    object = nb_fit,
    newdata = bos[!bos$train3, ]
)
table(nb_preds, bos$crimeful[!bos$train3]) # Confusion Matrix
```

```
##
## nb_preds FALSE TRUE
##    FALSE    41   14
##    TRUE      7   44
```

```
cat("Naive Bayes Model Accuracy: ",
    round(100*mean(qda_preds == bos$crimeful[!bos$train3]), 2), "%", sep = ""
)
```

```
## Naive Bayes Model Accuracy: 85.85%
```

Naive Bayes has approximately the same test accuracy as Quadratic Discriminant Analysis, which, again, could be due in part to sampling variability in the train-test split.

KNN Classifier

We'll now look at the performance of a KNN classifier on our data. We will look at a few different values of k:

```
train.X <- bos %>% filter(train3) %>% select(zn, nox, rad)
test.X <- bos %>% filter(!train3) %>% select(zn, nox, rad)
train.Y <- bos %>% filter(train3) %>% pull(crimeful)

knn_1_preds <- knn(train.X, test.X, train.Y, k=1)
knn_3_preds <- knn(train.X, test.X, train.Y, k=3)
knn_5_preds <- knn(train.X, test.X, train.Y, k=5)

cat("### K-Nearest Neighbor Model Accuracy ### \n",
    "K=1: ", round(100*mean(knn_1_preds == bos$crimeful[!bos$train3]), 2), "%\n",
    "K=3: ", round(100*mean(knn_3_preds == bos$crimeful[!bos$train3]), 2), "%\n",
    "K=5: ", round(100*mean(knn_5_preds == bos$crimeful[!bos$train3]), 2), "%\n",
    sep = ""
)
```

```
## ### K-Nearest Neighbor Model Accuracy ###
## K=1: 96.23%
## K=3: 96.23%
## K=5: 95.28%
```

The accuracy of the KNN classifier is extremely high compared to the other classification methods with this given set of predictors.