Linear & Quadratic Discriminant Analysis

Arnab Maity - Modified by Justin Post

library(MASS)  
library(klaR)  
library(nnet)  
library(glmnet)  
library(caret)  
library(rsample)  
library(ISLR2)  
library(knitr)  
library(tidyverse)  
library(mda)

We talked about two major ways to create classification models:

* Models that directly try to model the conditional class probabilities,
* Generative models that model ’s distribution and use Bayes’ theorem to obtain estimated conditional class probabilities

LDA & QDA take the generative model approach!

# Bayes’ Theorem

To understand how generative models work, let’s start with understanding Bayes’ theorem. Bayes’ theorem is built on the idea of conditional probability.

**Conditional Probability** - Let and be two events where . The conditional probability of A given B has occurred is

Assuming the , we can use conditional probability on to rewrite this as

This is essentially Bayes’ theorem!

Example: Suppose that we have three types of wines (cultivars). Call these and . Assume we have a measurement of quality on the wine (good or bad). We know, given the type of wine, the probability the wine is good:

Further, we know the relative occurrences of the three types of wine:

We can use Bayes’ theorem to find the probability the wine is of type given its good or bad status.

We just need to find the overall probability of good wine. This can be found using the law of total probability!

Thus, we have

The Bayes’ theorem can be extended to probability distributions (PDFs and PMFs if you are familiar with that language) rather than simply events like these.

In general, we have the following setup (I’m not going to use the book notation here as I think it is less clear than I’d like it to be):

* For -th item, we observe predictors $\X\_i = (X\_{i1}, \ldots, X\_{ip})^T$, and a class label (taking values in ).
* The conditional density function of is , .
* , such that . These are the *prior probabilities* of the classes.
  + In other words, probability that a randomly chosen observation comes from the prior -th class is .

By *Bayes theorem*, we obtain

The sample is predicted to be in the class that has the *highest posterior probability*. Notice that the denominator in the expression is same for any value of .

* The posterior probability is thus highest where the numerator is highest.

Of course, the density functions and the prior probabilities are unknown.

* We can estimate by using the relative proportion of its occurrence (assuming a random sample).
* where represents the number of observations in class .

For the unknown densities, we need to estimate them from the data! This can be modeled in a number of different ways.

# Linear & Quadratic Discriminant Analysis

LDA and QDA assume the conditional densities of are Normally distributed. The difference between LDA and QDA comes from the assumption on the variance of those Normal distributions.

These ideas are most easily shown through an example.

## Example: Developing QDA with One Predictor

To illustrate the basic ideas, consider the wines data set [available at the UCI machine learning repository](https://archive.ics.uci.edu/ml/datasets/wine).

Recall: The dataset contains quantities of 13 constituents found in each of the three types (cultivars) of wines.

* , but these numbers represent categories.

# Read the data  
wines <- read\_table("https://www4.stat.ncsu.edu/online/datasets/Wines.txt")  
# classes of wine  
table(wines$Class)

1 2 3   
59 71 48

We can easily approximate for .

p\_Y <- table(wines$Class)/nrow(wines)  
p\_Y

1 2 3   
0.3314607 0.3988764 0.2696629

Now we just need to obtain estimates for the conditional distributions of and we can use Bayes’ theorem to get a classifier!

* Let’s look at the relationships between our wine Class and some of the numeric variables in the data

|  |
| --- |
| Pairs-plot of a few variables of the wine data. |

Let’s just consider the Alcohol variable in creation of our model.

# new data set  
wine\_small <- wines |>  
 dplyr::select(Class, Alcohol) |>  
 mutate(Class = as.factor(Class))

We might consider the Normal distribution as a candidate for ’s distribution.

* We really have three distributions, one for each value can take on.

|  |
| --- |
| QQ plot for Alcohol for the two groups. |

* Plots show a fairly linear pattern
* Not unreasonable to assume that the data from all three classes follow normal distributions with possibly different means and variances.

We have:

* Three classes:
* One predictor: Alcohol content of a wine sample
* The conditional density function of is normal:

where we have possibly different means, , and variance,

The normal density function has the form

To estimate the densities, we need to estimate the unknown mean and variance parameters:

means\_variances <- wine\_small |>  
 group\_by(Class) |>  
 summarize(means = mean(Alcohol), vars = var(Alcohol))  
means\_variances

# A tibble: 3 × 3  
 Class means vars  
 <fct> <dbl> <dbl>  
1 1 13.7 0.214  
2 2 12.3 0.289  
3 3 13.2 0.281

With the assumption of normality, the estimated density functions are

as shown in the figure below.

|  |
| --- |
| Estimated distribution of Alcohol for the two groups. |

We can now combine this with our prior probabilities calculated earlier to obtain the values we need to make our classifications:

* For a given wine sample with Alcohol value, , we will classify the sample to group 1 if it has the largest value above
  + Suppose we have new data .

# new data  
x <- 12  
# density evaluated at x  
f <- dnorm(x,   
 mean = means\_variances$means,   
 sd = sqrt(means\_variances$vars))  
# p\_k \* f\_k  
pf <- p\_Y \* f  
round(pf, 4)

1 2 3   
0.0002 0.2586 0.0190

We see that Class 2 is our predicted class

* The posterior probabilities are

post\_prob <- pf / sum(pf)  
round(post\_prob,3)

1 2 3   
0.001 0.931 0.068

Reconsidering the image showing the three Normal densities, we see that 12 falls most closely to the middle of Class 2’s distribution and furthest from Class 1.

|  |
| --- |
| Estimated distribution of Alcohol for the two groups. |

We can try to develop rules around this to understand the idea more clearly.

* We can see that we favor Class 1 over Class 2 if our (Alcohol) is greater than some value, call it .
* To see this, we see that we favor Class 1 if . This is equivalent to favoring class 1 if
* Substituting the functional form of , we can rewrite the condition above as
* Define the functions
* Thus our classification rule is equivalent to assigning to class 1 if
* and
* The functions are called *discriminant functions*.

Now we can determine the decision boundaries with these functions. For instance, we can determine the value so that

(where is smaller)

* Solving this requires solving quadratic equations!
* Assume these values are larger than , then solving gives us
* This classification method is called **Quadratic Discriminant Analysis (QDA)**.
  + The name is due to the fact that the discriminant functions, are quadratic polynomials of .
  + Let’s see the decision boundary here.

qda <- train(Class ~ Alcohol,  
 data = wine\_small,  
 method = "qda")  
Alcohol <- seq(10, 16, length.out = 501)  
preds <- predict(qda, newdata = data.frame(Alcohol))

|  |
| --- |
| Estimated distribution of Alcohol for the two groups. |

* If we assumed the variances were the same across the Normal densities, the discriminant functions would be linear in . This procedure is called **Linear Discriminant Analysis**

## Example: QDA with Multiple Predictors

Now let us consider the case where we have two predictors (): and

* To extend QDA, we need to generalize the normal distribution to two-dimensions. We call such a distribution a *bivariate normal distribution*.

### Bivariate Normal Distribution

* Denote . is a *random vector*.
  + We can define the mean as :
  + To define the variance of the random vector we need to look at their individual variances, and , as well as their covariance, .
  + In general, we use the *variance-covariance matrix* of to summarize the variability of :

**Bivariate Normal Distribution**:

* The random vector follows a bivariate normal (Gaussian) distribution with mean vector and variance-covariance (positive definite) matrix if its joint PDF is given by

This is denoted by

The plots below give an example of this joint distribution and an example of a random sample from it.

|  |
| --- |
| Density function of a bivariate normal distribution (left) and a scatterplot of a random sample of size 100 from a bivariate normal distribution. |

The shape of the PDF (and that of the scatterplot of a random sample generated from the distribution) is determined by .

* An easy was to visualize the PDF of a bivariate distribution is to plot the constant probability density contours.

**Constant probability density contours**

We define the constant probability density contour (also called constant-density contour) of a bivariate normal PDF to be the set of vectors such that is constant, These sets are ellipses that are centered around .

The figures below show three examples of bivariate normal distribution with different variance-covariance patterns.

Note:

* If the two variables are uncorrelated (covariance is 0), the major and minor axes of the elliptical contours will be parallel to the - and -axis.
* In presence of correlation, the ellipses will be oriented according the sign/magnitude of the correlation.

|  |
| --- |
| PDF and contours of a bivariate normal distribution with . The contours are concentric circles since and are uncorrelated, and have the same variance. |

|  |
| --- |
| PDF and contours of a bivariate normal distribution with . The countours are oriented accoring to the positive correlation between and . Also, the contours are narrower along axis comapred to due to being more that . |

|  |
| --- |
| PDF and contours of a bivariate normal distribution with v(x1) = 1, v(x2) = 1.3, cov(x1,x2) = -0.5 |

### Multivariate Normal Distribution

Of course there is a general multivariate version of this distribution.

* A random vector is said for follow a multivariate normal distribution , where is a vector and is positive definite matrix, if the PDF of is
* We can show that and that .

### QDA with Predictors

To develop a classifier with predictors, look at the conditional distribution of the random vector (containing the predictors)

where and are the mean vector and variance-covariance matrix corresponding to class .

* As with our univariate predictor case, the true values of and are unknown, we estimate these parameters as

$$
\hat\boldsymbol{\mu}\_k = \mbox{ sample mean of group $k$ }
$$

$$
\hat \boldsymbol{\Sigma} = \mbox{ sample variance-covariance matrix of group $k$}
$$

The rest of the process is exactly as before!

* We need values for our ‘prior’ distributions,
* We find the value of
* We classify a new observation with data to class if the estimated posterior probability of class is highest, or equivalently, if is highest.

The discriminant functions are quadratic in each of the predictors in and are given by

* As before, an equivalent classification rule can be constructed using $\widehat\delta\_k(\x)$:
  + Assign to class where is largest among

This is the form of QDA for multiple predictors!

Of course we implement this using software!

### QDA Example Implementation

Let us consider the wines data again with all three classes, and two predictors, Alcohol and Proline.

* First let’s do some of this ourselves!
  + The estimated mean vectors and variance-covariance matrices for the conditional bivariate normal distributions are shown below.

X <- cbind(wines$Alcohol, wines$Proline)  
mu <- vector("list")  
Sigma <- vector("list")  
for(ii in 1:3){  
 mu[[ii]] <- colMeans(X[wines$Class == ii, ])  
 Sigma[[ii]] <- cov(X[wines$Class == ii, ])  
}  
mu

[[1]]  
[1] 13.74475 1115.71186  
  
[[2]]  
[1] 12.27873 519.50704  
  
[[3]]  
[1] 13.15375 629.89583

Sigma

[[1]]  
 [,1] [,2]  
[1,] 0.2135598 36.91949  
[2,] 36.9194944 49071.45003  
  
[[2]]  
 [,1] [,2]  
[1,] 0.2894055 3.651366  
[2,] 3.6513662 24715.367807  
  
[[3]]  
 [,1] [,2]  
[1,] 0.2811559 -5.434707  
[2,] -5.4347074 13247.329344

+ The figure below shows the data for the three classes, overlayed with bivariate normal density contours for each class.

|  |
| --- |
| Three class wines data overlayed with normal density contours. |

+ As before, \*\*for a given data point\*\* $\boldsymbol{x}$, we can compute $\widehat p\_Y(k) \widehat f\_{\boldsymbol{X}|Y}(\boldsymbol{x}|y=k)$, and the associated posterior probabilities.

# For multivariate normal density  
library(mnormt)  
# new data  
newx <- data.frame(Alcohol = 13, Proline = 600)  
# p-hat  
p <- table(wines$Class)/nrow(wines)  
# f-hat  
f <- c()  
for(ii in 1:3){  
 f[ii] <- dmnorm(newx, mean = mu[[ii]], varcov = Sigma[[ii]])  
}  
# Posterior prob  
post\_prob <- p\*f / sum(p\*f)  
round(post\_prob, 3)

1 2 3   
0.027 0.290 0.683

* In general, we can use the qda() function in MASS package to build QDA models.

wine\_qda <- qda(Class ~ Alcohol + Proline, data = wines)  
wine\_qda

Call:  
qda(Class ~ Alcohol + Proline, data = wines)  
  
Prior probabilities of groups:  
 1 2 3   
0.3314607 0.3988764 0.2696629   
  
Group means:  
 Alcohol Proline  
1 13.74475 1115.7119  
2 12.27873 519.5070  
3 13.15375 629.8958

# prediction of new x  
pred <- predict(wine\_qda, newdata = newx)  
pred

$class  
[1] 3  
Levels: 1 2 3  
  
$posterior  
 1 2 3  
1 0.027429 0.2898352 0.6827358

The decision boundaries for this QDA are shown in the figure below.

* Here the boundaries are quadratic functions of Alcohol and Proline since the discriminant functions are quadratic!

|  |
| --- |
| Decision boundary of QDA when applied to wines data. |

We can also estimate the test error rate of QDA when applied to wine data using data splitting methods such as CV or holdout. We can use caret to do so.

set.seed(1001)  
caret\_qda <- train(factor(Class) ~ Alcohol + Proline,  
 data = wines,  
 method = "qda",  
 trControl = trainControl(method = "CV",  
 number = 10))  
caret\_qda$results

parameter Accuracy Kappa AccuracySD KappaSD  
1 none 0.8291667 0.7402777 0.1446417 0.2199071

confusionMatrix(caret\_qda)

Cross-Validated (10 fold) Confusion Matrix   
  
(entries are percentual average cell counts across resamples)  
   
 Reference  
Prediction 1 2 3  
 1 29.8 0.6 1.1  
 2 0.0 34.8 7.3  
 3 3.4 4.5 18.5  
   
 Accuracy (average) : 0.8315

## Different Clasification Methods

Following similar logic to our QDA exploration, we could specify the densities in a different way, or put other assumptions on the parameters, to obtain other classification methods!

* For example, we might assume that the multivariate normal distributions have the *same variance-covariance matrix*, , in each class.
  + Then it can be shown that the discriminant functions are *linear* in .
  + The common variance-covariance matrix, , can be estimated by a “pooled” estimator:
  + where are sample covariance matrices of ’s from class , respectively.

The corresponding classification method is known as *Linear Discriminant Analysis (LDA)*, and the decision boundaries are linear!

In general, various classification methods specify or estimate the densities in different ways, giving rise to different classification rules. Some of these methods are shown below:

* **Linear discriminant analysis** (LDA) (lda function in MASS library): use Gaussian densities with different means but *same* covariance matrix for each class
* **Quadratic discriminant analysis** (QDA) (qda function in MASS library): use Gaussian densities with different means and *different* covariance matrices for each class
* **Naive Bayes Classifier** (NaiveBayes function in klaR library): uses estimated density assuming that the inputs are conditionally independent in each class
* **Regularized Discriminant Analysis** (rda function in klaR library): using regularized group covariance matrices that are robust against multicollinearity in the data
* **Flexible discriminant analysis** (fda function in mda library): regerssion based classifier, captures nonlinear features of the covariates
* **Mixture discriminant analysis** (mda function in mda library): density of each class is modeled using a *mixture* (weighted sum) of normal densities, can model multimodal densities
* **Kernel Density Classification** (kda function in ks library): densities are estimated nonparametrically using kernel density estimation.

The figure below shows the decision boundaries of a few classification methods applied to wines data.

* Keep in mind that there are *many* more discrimination analysis methods available in literature and in various R and python packages!

|  |
| --- |
| Classification boundaries for four classifiers. |

## LDA vs. QDA

Even though QDA can be considered more general method that LDA (due to the restrictive assumption made in LDA that each class has the same variance-covariance matrix), we might still prefer LDA in some situations over QDA.

* QDA requires estimation of a larger number of parameters than LDA.
  + In our wine data example, suppose we use all predictors, with classes.
  + QDA estimates variance-covariance matrix with size .
  + Thus QDA estimates a total of parameters!
  + In comparison, LDA requires estimation of only one common variance-covariance matrix.
  + Since LDA is a less flexible model, it may have more bias but less variance. Thus sometimes LDA might have better prediction performance than QDA.

Roughly speaking, LDA tends to be a better choice than QDA if training sample size is small and so reducing variance is crucial. In contrast, QDA is recommended if the training set is very large, so that the variance of the classifier is not a major concern, or if the assumption of a common covariance matrix for the classes is clearly untenable.

|  |
| --- |
| LDA vs QDA in two simulated example. Left panel shows a data set where each class has the same covariance matrix. The right panel shows a data set with different covariance matrices for each class. The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries are displayed. The shading indicates the QDA decision rule |

The figure above shows examples of simulated data sets – figure taken from *Introduction to Statistical Learning*.

* Each of the data sets contains two variables and and two classes.
* The left panel shows the data with for both the classes. Thus the Bayes decision boundary is linear in this case, and LDA performs better than QDA, because QDA suffers from higher variance without a corresponding decrease in bias.
* The right panel shows a data set with in one class and in the other. Here, the assumption of a common variance-covariance matrix is not appropriate and thus, LDA suffers from high bias but QDA performs better.

A possible disadvantage of QDA/LDA is the assumption of multivariate normality of predictors in each class. When this assumption is unreasonable LDA/QDA can perform badly. However, assumption of normality is more crucial for QDA than LDA since another formulation of LDA does not require normality of the predictors.

## Naive Bayes classifier

Recall that LDA and QDA estimate the densities for the classes using the multivariate normality assumption.

The naive Bayes classifier makes a single simplifying assumption:

* The joint density of conditional on is the product of individual conditional density functions,
* In other words, within each class the predictors are assumed to be *independent*.
* This is a quite strong assumption – this will also imply that there is *no relation* (linear or otherwise) between the predictors *within each class*.
* In most situations, this assumption is not appropriate.
* However, a classifier can still be constructed based on this assumption, and it often gives good results (especially for smaller )!

With this assumption, we only need to form estimates of the *marginal conditional density functions*, , to obtain an estimate of :

The rest of the procedure is the same as before:

* We compute the posterior probabilities, or equivalently , and classify observations accordingly

We can form by following any of the options below:

* If is quantitative, we can assume .
  + Then we only need to estimate and .
  + This is equivalent to running QDA with a
* Another option for quantitative predictors is to estimate the densities with nonparametric methods.
  + Examples of such estimators are *relative frequency histograms* and *kernel density estimators* – a smoothed version of histogram.

|  |
| --- |
| Relative frequancy histogram and kernel density estimator (solid line) of a sample. |

* If is qualitative, then, conditional on the class , we can simply take the proportion of sample observations for each value of the predictor.
  + In other words, is the estimated *probability mass function* of the -th predictor:

$$
\widehat f\_{X\_j|Y}(x\_j|y=k) = \frac{1}{n\_{\rm train, k}} \sum\_i I(X\_{ij} = x\_j),
$$

* + where $n\_{\rm train, k}$ is the size of the training set for the -th class.

In R, we can use the NaiveBayes() function in the klaR library for build a naive Bayes classifier.

library(klaR)  
nb\_wine <- NaiveBayes(as.factor(Class) ~ Alcohol + Proline,   
 data = wines,  
 usekernel = FALSE)  
nb\_kern <- NaiveBayes(as.factor(Class) ~ Alcohol + Proline,   
 data = wines,  
 usekernel = TRUE)  
predict(nb\_wine, newdata = data.frame(Alcohol = 13,  
 Proline = 600))

$class  
[1] 3  
Levels: 1 2 3  
  
$posterior  
 1 2 3  
[1,] 0.01007071 0.2884064 0.7015229

predict(nb\_kern, newdata = data.frame(Alcohol = 13,  
 Proline = 600))

$class  
[1] 3  
Levels: 1 2 3  
  
$posterior  
 1 2 3  
[1,] 0.02208289 0.2097631 0.768154

* In the first fit, we assumed normal distribution for each predictor (usekernel = FALSE)
* In the second fit, we use kernel density estimation (usekernel = TRUE).

We can also use caret to evaluate test error with method = nb!

Below we fit an LDA, QDA, and two forms of Naive bayes to a simple model for the wine data.

tr <- trainControl(method = "repeatedcv", number = 10, repeats = 25)  
lda <- train(as.factor(Class) ~ Alcohol + Proline,   
 data = wines,  
 method = "lda",  
 trControl = tr)  
qda <- train(as.factor(Class) ~ Alcohol + Proline,  
 data = wines,  
 method = "qda",  
 trControl = tr)  
nb\_g <- train(as.factor(Class) ~ Alcohol + Proline,   
 data = wines,  
 method = "nb",   
 tuneGrid = expand.grid(usekernel = FALSE,  
 fL = 0,  
 adjust = 1),  
 trControl = tr)  
nb\_k <- train(as.factor(Class) ~ Alcohol + Proline,   
 data = wines,  
 method = "nb",   
 tuneGrid = expand.grid(usekernel = TRUE,  
 fL = 0,  
 adjust = 1),  
 trControl = tr)

Here we can plot the accuracy and kappa values across the repeated test sets.

bwplot(resamples(list(LDA = lda,   
 QDA = qda,   
 NaiveB\_gauss = nb\_g,  
 NaiveB\_kern = nb\_k)))

|  |
| --- |
| Accuracy and Kappa values for the four models on the wine data set. |

# Comparison of a few classifiers

Chapter 4.5 of the textbook provides a nice comparison of the different classifiers we learned. I recommend you read the chapter to get more insight. We will provide only a brief overview of the discussion presented in the book.

Analytically, we can compare the form of log-odds of LDA, QDA, naive Bayes and logistic regression. We can show that:

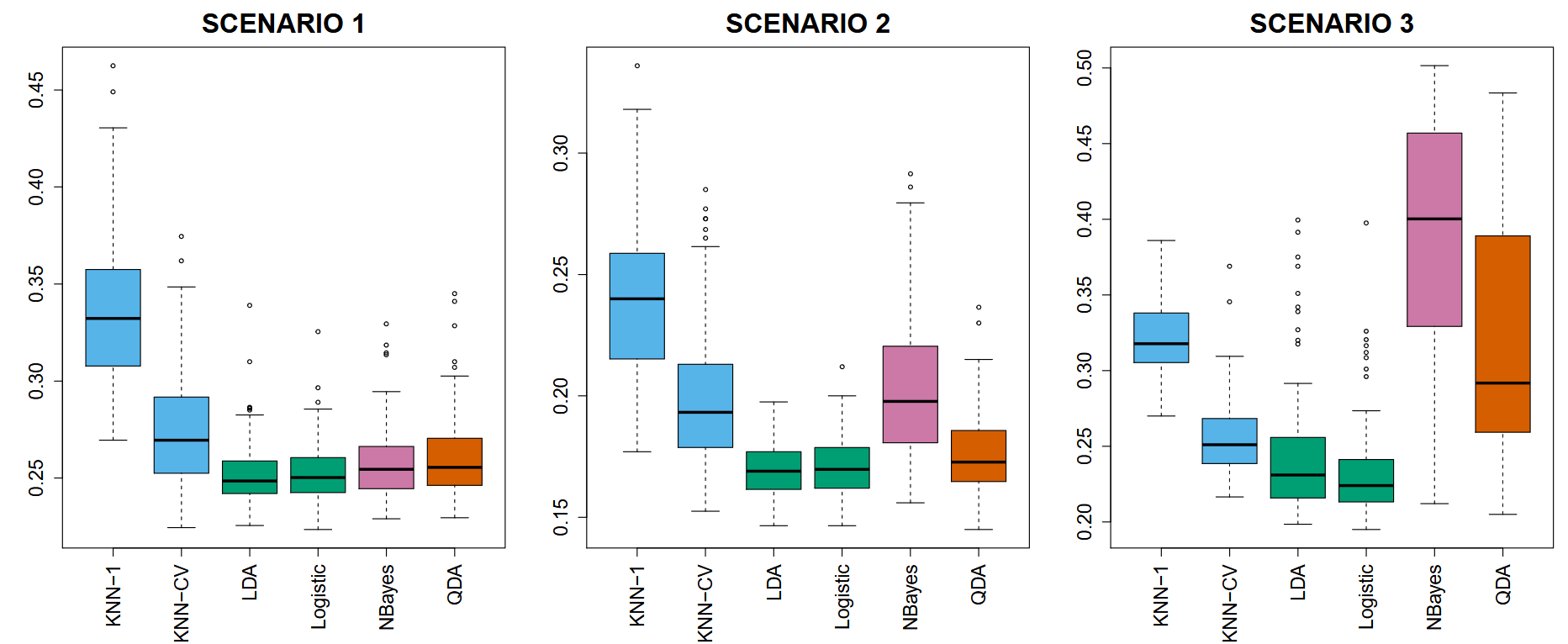
* LDA and logistic regression models the log-odds as a linear combination of the predictors.
* QDA models the log-odds as a quadratic function of the predictors.
* Naive Bayes classifier model the log-odds as a sum of non-linear functions of the predictors.

Any classifier with a linear decision boundary is a special case of naive Bayes.

However, neither QDA nor naive Bayes is a special case of the other.

Between LDA and logistic regression, we expect LDA to perform better if the assumptions for LDA are satisfied.

The textbook also provides numerical comparison between various classifiers discussed so far. The figure below shows test error rates under different scenarios – see the textbook for details for each scenarios.



|  |
| --- |
| Test error rates for a few classifiers in linear (top row) and nonlinear (bottom row) scenarios described in , section 4.5. |

In general, no single classifier performs best in every scenario. Their test performance depends on the underlying structure (distribution, variance-covariance patterns) of the the data.

* If it often a good idea to build several classifiers, and evaluate them using their test error rate!