STATS 415 Lab4

Weijing Tang

Feb 1, 2018

1 Today's objectives

1. Recall data visualization and how to split training and test data.
2. Practice how to implement LDA and QDA and calculate test and training error.
3. If time permitted, exercise to perform LDA and QDA step by step.

2 Data and necessary packages

In this lab, we will use Iris data to perform LDA and QDA. Recall that **Species** is a three-level categorical variable. We will use it as the target of classification.

```
data(iris)
```

3 Test-train Split

The estimated priors of LDA and QDA is the proportion of each class, which we will go over later. So to remove unnecessary variance, we should split each class proportionately for training and testing.

```
set.seed(430)
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

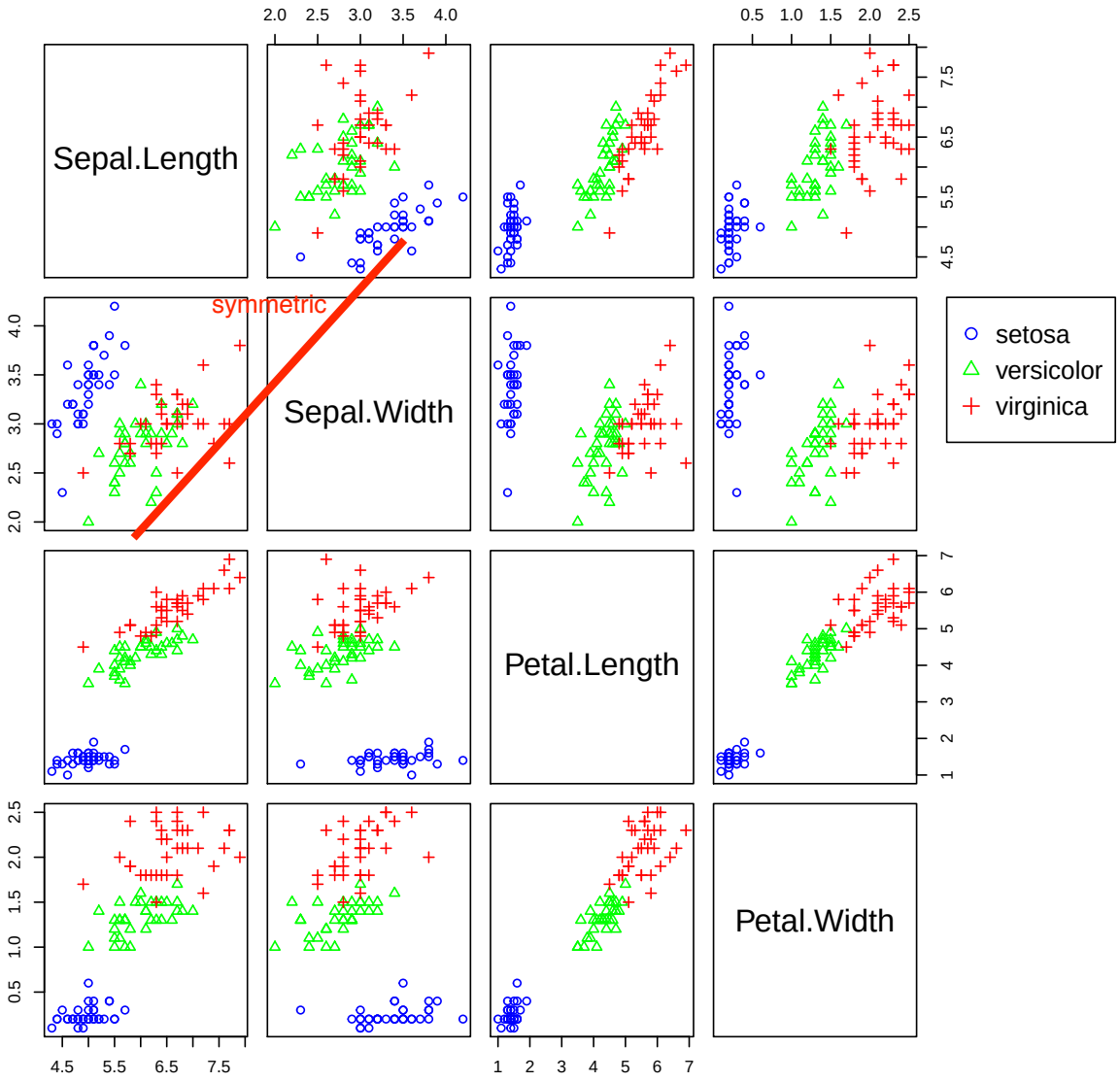
```
iris_setosa = which(iris$Species == "setosa")
iris_versi = which(iris$Species == "versicolor")
iris_virg = which(iris$Species == "virginica")
train_id = c(sample(iris_setosa, size = trunc(0.70 * length(iris_setosa))),
             sample(iris_versi, size = trunc(0.70 * length(iris_versi))),
             sample(iris_virg, size = trunc(0.70 * length(iris_virg))))
iris_train = iris[train_id, ]
iris_test = iris[-train_id, ]
```

4 Data Visualization

Let's explore the training data graphically to have a glance at the relationship between **Species** and the other variables. Remember that our target is **Species**, we can use different colors or symbols to distinguish them in the plots.

4.1 Scatter Plots

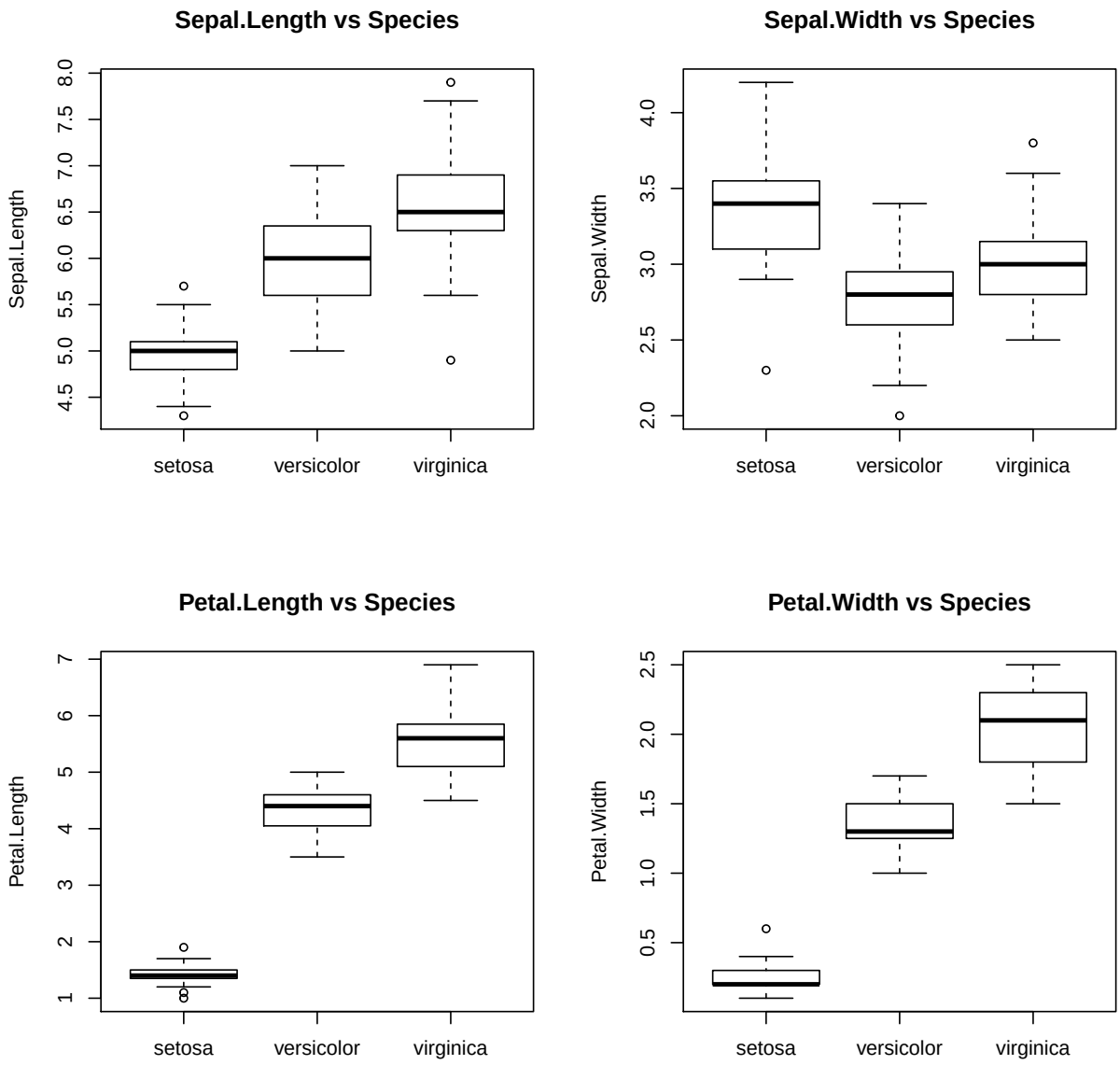
```
pairs(iris_train[1:4], col=c("blue", "green","red")[iris_train$Species],
      oma=c(4,4,6,12), pch=c(1,2,3)[iris_train$Species])
par(xpd=TRUE)
legend(0.85, 0.7, as.vector(unique(iris_train$Species)),
       col=c("blue", "green", "red"), pch=1:3)
```



4.2 Side-by-side Boxplots

```
par(mfrow=c(2,2))
plot(iris_train$Species,iris_train$Sepal.Length, main = "Sepal.Length vs Species",
```

```
     ylab = "Sepal.Length")
plot(iris_train$Species,iris_train$Sepal.Width, main = "Sepal.Width vs Species",
     ylab = "Sepal.Width")
plot(iris_train$Species,iris_train$Petal.Length, main = "Petal.Length vs Species",
     ylab = "Petal.Length")
plot(iris_train$Species,iris_train$Petal.Width, main = "Petal.Width vs Species",
     ylab = "Petal.Width")
```



Question: Which of features seem most likely to be useful in predicting **Species**?

petal length and petal width

5 Generative Models

In this lab, we continue our discussion of classification methods. We will implement new methods, LDA and QDA, each a **generative** method.

Generative methods model the joint probability, $p(x, y)$, often by assuming some distribution for the conditional distribution of X given Y, $f(x \mid y)$. Bayes theorem is then applied to classify according to $p(y \mid x)$.

LDA and QDA will use Bayes theorem to build a classifier.

$$p_k(x) = P(Y = k \mid X = x) = \frac{\pi_k \cdot f_k(x)}{\sum_{g=1}^{G} \pi_g \cdot f_g(x)}$$

to maximize posterior probability

We call $p_k(x)$ the **posterior** probability, which we will estimate then use to create classifications. The π_g are called the **prior** probabilities for each possible class g. That is, $\pi_g = P(Y = g)$, unconditioned on X. The $f_g(x)$ are called the **likelihoods**, which are indexed by g to denote that they are conditional on the classes. The denominator is often referred to as a **normalizing constant**.

Classifications are made to the class with the highest estimated posterior probability, which is equivalent to the class with the largest

$$\hat{p}_k(x) \propto \hat{\pi}_k \cdot \hat{f}_k(\mathbf{x}).$$

6 Linear Discriminant Analysis

LDA assumes that the predictors are multivariate normal conditioned on the classes.

$$X \mid Y = k \sim N(\mu_k, \Sigma)$$

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu_k)'\Sigma^{-1}(\mathbf{x} - \mu_k)\right]$$

Notice that Σ does **not** depend on k, that is, we are assuming the same Σ for each class. We then use information from all the classes to estimate Σ.

To fit an LDA model, we use the `lda()` function from the `MASS` package.

```
library(MASS)
iris_lda = lda(Species ~ ., data = iris_train)
iris_lda
```

```
## Call:
## lda(Species ~ ., data = iris_train)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.965714    3.365714     1.434286   0.2428571
## versicolor     6.000000    2.765714     4.308571   1.3285714
## virginica      6.594286    3.002857     5.545714   2.0714286
```

```
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  1.224439  0.05864559
## Sepal.Width   1.150894  1.96257722
## Petal.Length -2.805008 -1.47454038
## Petal.Width  -2.235457  3.85600160
##
## Proportion of trace:
##    LD1    LD2
## 0.9877 0.0123
```

1/3

Question: Where are the estimated $\hat{\pi}_k$ and $\hat{\mu}_k$ for each class?

The `predict()` function operates in a new way when called on an `lda` object. By default, it returns an entire list. Within that list `class` stores the classifications and `posterior` contains the estimated probability for each class.

```
names(predict(iris_lda, iris_train))
```

```
## [1] "class"     "posterior" "x"
```

```
head(predict(iris_lda, iris_train)$class, n = 9)
```

```
## [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

```
head(predict(iris_lda, iris_train)$posterior, n = 9)
```

```
##    setosa   versicolor    virginica
## 8       1 3.096005e-22 1.426933e-43
## 35      1 6.401627e-20 8.147268e-41
## 12      1 6.817195e-20 2.153361e-40
## 13      1 7.625870e-21 1.638336e-42
## 46      1 2.604936e-19 1.163677e-39
## 48      1 3.304543e-20 6.184837e-41
## 37      1 4.785555e-28 2.403647e-51
## 5       1 1.042476e-24 1.339372e-46
## 28      1 6.845302e-24 9.827451e-46
```

6.1 Training and test error

We store the predictions made on the training and test sets.

```
iris_lda_train_pred = predict(iris_lda, iris_train)$class
iris_lda_test_pred = predict(iris_lda, iris_test)$class
```

We write a function named **calc_class_err** to calculate error. (Recall **mse** in Lab3.)

```
calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}
```

```
calc_class_err(predicted = iris_lda_train_pred, actual = iris_train$Species)
```

```
## [1] 0.00952381
```

```
calc_class_err(predicted = iris_lda_test_pred, actual = iris_test$Species)
```

```
## [1] 0.04444444
```

```
table(predicted = iris_lda_test_pred, actual = iris_test$Species)
```

```
##             actual
## predicted    setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         0
##   virginica       0          2        15
```

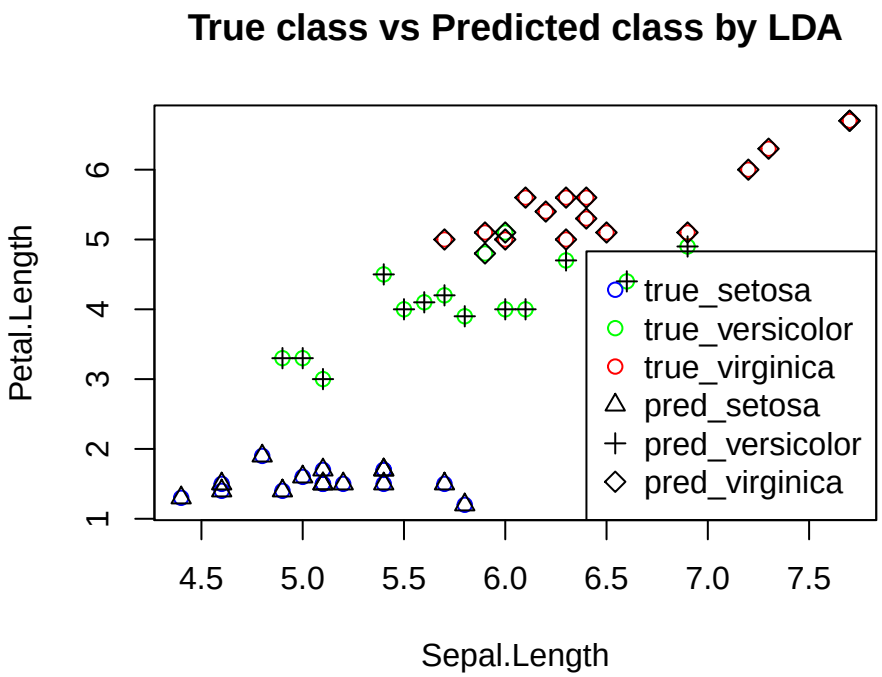
Question: How does LDA perform?

6.2 Visualize results

```
plot(iris_test$Sepal.Length,iris_test$Petal.Length,
     col = c("blue", "green","red")[iris_test$Species],
     xlab = "Sepal.Length", ylab = "Petal.Length",
     main = "True class vs Predicted class by LDA"
     )
points(iris_test$Sepal.Length,iris_test$Petal.Length,
     pch = c(2,3,5)[iris_lda_test_pred])
```

three different symbols for the three species

```
legend("bottomright", c("true_setosa","true_versicolor","true_virginica",
                        "pred_setosa","pred_versicolor","pred_virginica"),
       col=c("blue", "green", "red", "black", "black", "black"),
       pch=c(1,1,1,2,3,5))
```



6.3 Pre-specified Priors

Instead of learning (estimating) the proportion of the three species from the data, we could instead specify them ourselves. Here we choose a nonuniform distributions over the possible species.

```
iris_lda2 = lda(Species ~ ., data = iris_train, prior = c(4, 1, 1) / 6)
iris_lda2
```

for train error

data=iris_test for test error

```
## Call:
## lda(Species ~ ., data = iris_train, prior = c(4, 1, 1)/6)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.6666667  0.1666667  0.1666667
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.965714    3.365714     1.434286   0.2428571
## versicolor     6.000000    2.765714     4.308571   1.3285714
## virginica      6.594286    3.002857     5.545714   2.0714286
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length -1.225221  0.03901686
## Sepal.Width  -1.182196  1.94388254
## Petal.Length  2.828277 -1.42940177
## Petal.Width   2.173379  3.89132890
##
## Proportion of trace:
```

```
##    LD1    LD2
## 0.9933 0.0067
```

```
iris_lda_train_pred2 = predict(iris_lda2, iris_train)$class
iris_lda_test_pred2 = predict(iris_lda2, iris_test)$class
```

```
calc_class_err(predicted = iris_lda_train_pred2, actual = iris_train$Species)
```

```
## [1] 0.00952381
```

```
calc_class_err(predicted = iris_lda_test_pred2, actual = iris_test$Species)
```

```
## [1] 0.04444444
```

```
table(predicted = iris_lda_test_pred2, actual = iris_test$Species)
```

```
##             actual
## predicted    setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         0
##   virginica       0          2        15
```

This actually gives the same test accuracy.

7 Quadratic Discriminant Analysis

QDA also assumes that the predictors are multivariate normal conditioned on the classes.

$$X \mid Y = k \sim N(\mu_k, \Sigma_k)$$

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_k)'\Sigma_k^{-1}(\mathbf{x}-\mu_k)\right]$$

Notice that now Σ_k **does** depend on k, that is, we are allowing a different Σ_k for each class. We only use information from class k to estimate Σ_k.

Like `lda()`, the `qda()` function is found in the `MASS` package.

```
iris_qda = qda(Species ~ ., data = iris_train)
iris_qda
```

```
## Call:
## qda(Species ~ ., data = iris_train)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.965714    3.365714     1.434286   0.2428571
## versicolor     6.000000    2.765714     4.308571   1.3285714
## virginica      6.594286    3.002857     5.545714   2.0714286
```

Here the output is similar to LDA, again giving the estimated $\hat{\pi}_k$ and $\hat{\mu}_k$ for each class.

The `predict()` function operates the same as the `predict()` function for LDA.

```
iris_qda_train_pred = predict(iris_qda, iris_train)$class
iris_qda_test_pred = predict(iris_qda, iris_test)$class
```

```
calc_class_err(predicted = iris_qda_train_pred, actual = iris_train$Species)
```

```
## [1] 0
```

The training error is 0.think of overfit if test error is large

```
calc_class_err(predicted = iris_qda_test_pred, actual = iris_test$Species)
```

```
## [1] 0.04444444
```

```
table(predicted = iris_qda_test_pred, actual = iris_test$Species)
```

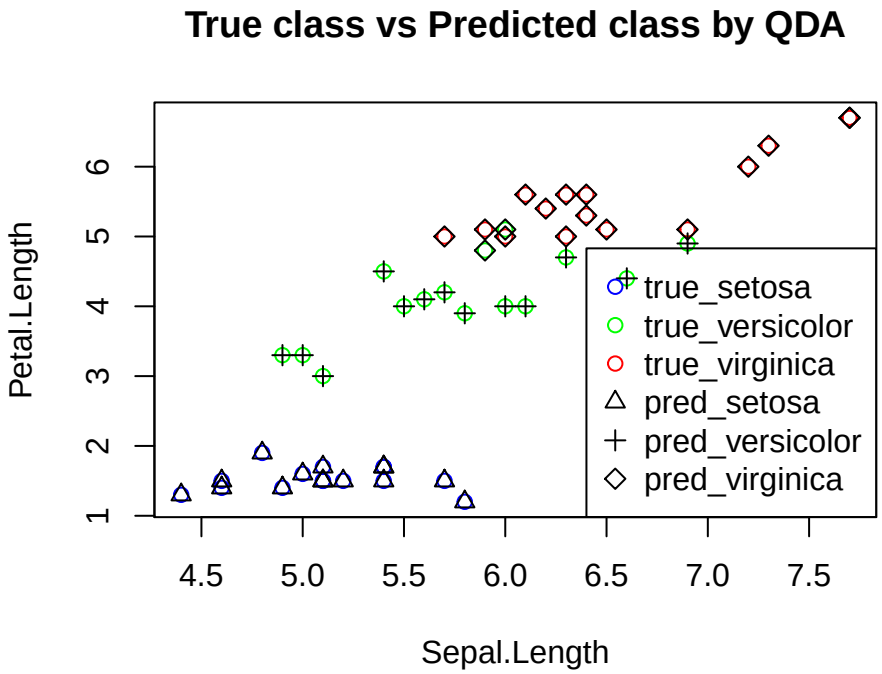
```
##             actual
## predicted    setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         0
##   virginica       0          2        15
```

Question: How does QDA perform?

7.1 Visualize results

```
plot(iris_test$Sepal.Length,iris_test$Petal.Length,
     col = c("blue", "green","red")[iris_test$Species],
     xlab = "Sepal.Length", ylab = "Petal.Length",
     main = "True class vs Predicted class by QDA"
     )
points(iris_test$Sepal.Length,iris_test$Petal.Length,
     pch = c(2,3,5)[iris_qda_test_pred])

legend("bottomright", c("true_setosa","true_versicolor","true_virginica",
                        "pred_setosa","pred_versicolor","pred_virginica"),
       col=c("blue", "green", "red", "black", "black", "black"),
       pch=c(1,1,1,2,3,5))
```



Method	Train Error	Test Error
LDA	0.0095238	0.0444444
LDA, Nonuniform Prior	0.0095238	0.0444444
QDA	0.0000000	0.0444444

Question: What if compared with LDA?

If the performance of LDA and QDA are the same, choose LDA because it's a simpler model with fewer parameters. In this case, we may prefer QDA.

8 Exercise*

Suppose our data $(x_i, y_i), i = 1, \cdots, n_0 + n_1$ are generated from the following model:

for $i = 1, \cdots, n_0$,

$$y_i = 0, \quad x_i | y_i = 0 \sim N(\mu_0, \sigma_0^2)$$

for $i = n_0 + 1, \cdots, n_0 + n_1$

$$y_i = 1, \quad x_i | y_i = 1 \sim N(\mu_1, \sigma_1^2)$$

8.1 Question: How to derive the LDA and QDA rules?

The LDA and QDA classifiers are defined by optimizing

$$\widehat{C}(x) = \arg\max_{k=0,1} P(y = k|x)$$

Recall that

$$P(y = k|x) \propto P(x|y = k)P(y = k)$$

where for LDA,

$$P(x|y = k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x - \mu_k)^2}{2\sigma^2}),$$

we assume the variance is the same between two classes. So here σ^2 is the pooled covariance

$$\sigma^2 = \frac{(n_0 - 1)\sigma_0^2 + (n_1 - 1)\sigma_1^2}{n_1 + n_0 - 2}.$$

And for QDA,

$$P(x|y = k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp(-\frac{(x - \mu_k)^2}{2\sigma_k^2}).$$

The classifiers can be simplified as optimizing over the discriminant function δ_k so that for a given value x_0 the classifier is defined as

$$\widehat{C}(x_0) = \arg\max_{k=0,1} \delta_k(x_0)$$

where for LDA

$$\delta_k(x) = \frac{x\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k$$

and for QDA

$$\delta_k(x) = -\log(\sigma_k) - \frac{(x - \mu_k)^2}{2\sigma_k^2} + \log \pi_k$$

Here we estimate $\pi_0 = \frac{n_0}{n_0+n_1}$ and $\pi_1 = \frac{n_1}{n_0+n_1}$.

For LDA, we predict the class of y as 1 only if

$$\frac{x\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + \log \pi_1 > \frac{x\mu_0}{\sigma^2} - \frac{\mu_0^2}{2\sigma^2} + \log \pi_0$$

i.e. if sigma is the same, the quadratic term can be moved so it's LDA

$$x(\mu_1 - \mu_0) > \mu_1^2/2 - \mu_0^2/2 + \sigma^2 \log \frac{n_0}{n_1}$$

$$x > \frac{\mu_1^2/2 - \mu_0^2/2 + \sigma^2 \log \frac{n_0}{n_1}}{\mu_1 - \mu_0}$$

which is actually solving a linear inequality.

For QDA, we predict the class of y as 1 only if

$$\text{-log(2π)^0.5} - \log(\sigma_1) - \frac{(x - \mu_1)^2}{2\sigma_1^2} + \log \pi_1 > -\log(\sigma_0) - \frac{(x - \mu_0)^2}{2\sigma_0^2} + \log \pi_0 \quad \text{-log(2π)^0.5}$$

which is solving a quadradic inequality.

8.2 Question: Given $\mu_0 = 2.5, \mu_1 = -1, \sigma_0^2 = 1, \sigma_1^2 = 2$ and $n_0 = 100, n_1 = 150$, what is the test error of the following test dataset?

x	4	3.8	0.5	0	-5
y	0	0	1	1	1

You can use R as a calculator.

9 Resources and References

1. https://daviddalpiaz.github.io/r4sl/generative-models.html#linear-discriminant-analysis