```r
# -------------------------------------------------------------------------------
#
# Generative Models - qda and lda
#
# -------------------------------------------------------------------------------

# Setup
library(tidyverse)
library(dslabs)
library(dplyr)
library(ggplot2)
library(Lahman)
library(HistData)
library(caret)
library(e1071)
library(matrixStats)

#
# QDA
#

# Quadratic discriminant analysis, or QDA, is a version of Naive Bayes in which
# we assume that the conditional probabilities for the predictors are
# multivariate normal. So the simple example we described in our Naive Bayes
# video was actually QDA.

# In this video, we're going to look at a slightly more complicated example
# where we have two predictors. It's the 2 or 7 example that we've previously
# seen. We can load it with this code.

data("mnist_27")

# In this case, we have two predictors. So we assume that their conditional
# distribution is bivariate normal. This implies that we need to estimate two
# averages, two standard deviations, and a correlation for each case, the 7s and
# the 2s. Once we have these, we can approximate the conditional distributions.
# We can easily estimate these parameters from the data using this simple code.

params <- mnist_27$train %>%
     group_by(y) %>%
     summarize(avg_1 = mean(x_1), avg_2 = mean(x_2),
               sd_1= sd(x_1), sd_2 = sd(x_2),
               r = cor(x_1,x_2))
params
#> # A tibble: 2 x 6
#>   y     avg_1 avg_2   sd_1   sd_2     r
#>   <fct> <dbl> <dbl>  <dbl>  <dbl> <dbl>
#> 1 2     0.129 0.283 0.0702 0.0578 0.401
#> 2 7     0.234 0.288 0.0719 0.105  0.455

# We can also visually demonstrate the approach. We plot the data and use
# contour plots to give an idea of what the two estimated normal densities look
# like. We show a curve representing a region that includes 95% of the points.

mnist_27$train %>% mutate(y = factor(y)) %>%
     ggplot(aes(x_1, x_2, fill = y, color=y)) +
     geom_point(show.legend = FALSE) +
     stat_ellipse(type="norm", lwd = 1.5)

# Once you've estimated these two distributions, this defines an estimate for
# the conditional probability of y equals 1 given x1 and x2.

# We can the caret package to fit the model and obtain predictors. The code is
# quite simple and it looks like this.

library(caret)
train_qda <- train(y ~ .,
                   method = "qda",
                   data = mnist_27$train)

# We see that we obtain a relatively good accuracy of 0.82.

y_hat <- predict(train_qda, mnist_27$test)
confusionMatrix(data = y_hat, reference = mnist_27$test$y)$overall["Accuracy"]
#> Accuracy
#>     0.82

# The estimated conditional probability looks relatively similar to the true
# distribution, as we can see here.

# INSERT CODE FOR GRAPH

# Although the fit is not as good as the one we obtain with kernel smoothers,
# which we saw in a previous video. And there's a reason for this. The reason is
# that we can show mathematically that the boundary must be a quadratic function
# of the form x2 equals ax1 squared plus bx plus c. One reason QDA does not work
# as well as the kernel method is perhaps because the assumption of normality do
# not quite hold. Although for the 2s, the bivariate normal approximation seems
# reasonable, for the 7 it does seem to be off. Notice the slight curvature.

mnist_27$train %>% mutate(y = factor(y)) %>%
     ggplot(aes(x_1, x_2, fill = y, color=y)) +
     geom_point(show.legend = FALSE) +
     stat_ellipse(type="norm") +
     facet_wrap(~y)

# Although QDA work well here, it becomes harder to use as a number of
# predictors increases. Here we have two predictors and have to compute four
# means, four standard deviations, and two correlations. How many parameters
# would we have to estimate if instead of two predictors we had 10? The main
# problem comes from estimating correlations for 10 predictors. With 10, we have
# 45 correlations for each class. In general, this formula tells us how many
# parameters we have to estimate, and it gets big pretty fast. Once the number
# of parameters approaches the size of our data, the method becomes unpractical
# due to overfitting.

#
# lDA
#

# A relatively simple solution to the problem of having too many parameters is
# to assume that the correlation structure is the same for all classes, which
# reduces the number of parameters we need to estimate. In this case, we would
# compute just one pair of standard deviations and one correlation, so the
# parameters would look something like this:

params <- mnist_27$train %>%
     group_by(y) %>%
     summarize(avg_1 = mean(x_1), avg_2 = mean(x_2), sd_1= sd(x_1), sd_2 = sd(x_2), r = cor(x_1,x_2))

params <-params %>% mutate(sd_1 = mean(sd_1), sd_2=mean(sd_2), r=mean(r))
params
#> # A tibble: 2 x 6
#>   y     avg_1 avg_2   sd_1   sd_2     r
#>   <fct> <dbl> <dbl>  <dbl>  <dbl> <dbl>
#> 1 2     0.129 0.283 0.0710 0.0813 0.428
#> 2 7     0.234 0.288 0.0710 0.0813 0.428

# The distributions now look like this:

mnist_27$train %>% mutate(y = factor(y)) %>%
     ggplot(aes(x_1, x_2, fill = y, color=y)) +
     geom_point(show.legend = FALSE) +
     stat_ellipse(type="norm", lwd = 1.5)

# Now the size of the ellipses as well as the angle are the same. This is
# because they have the same standard deviations and correlations. When we force
# this assumption, we can show mathematically that the boundary is a line, just
# as with logistic regression. For this reason, we call the method linear
# discriminant analysis (LDA). Similarly, for QDA, we can show that the boundary
# must be a quadratic function.

# INSERT CODE FOR GRAPH

# In this case, the lack of flexibility does not permit us to capture the
# nonlinearity in the true conditional probability function. We can fit the
# model using caret:

train_lda <- train(y ~ .,
                   method = "lda",
                   data = mnist_27$train)
y_hat <- predict(train_lda, mnist_27$test)
confusionMatrix(data = y_hat, reference = mnist_27$test$y)$overall["Accuracy"]
#> Accuracy
#>     0.75

# ...and we can see that our accuracy is quite low.
```