UNIVERSITY OF WISCONSIN
DATA SCIENCE

DS 740

Data Mining

Linear Discriminant Analysis and Bayes' Rule

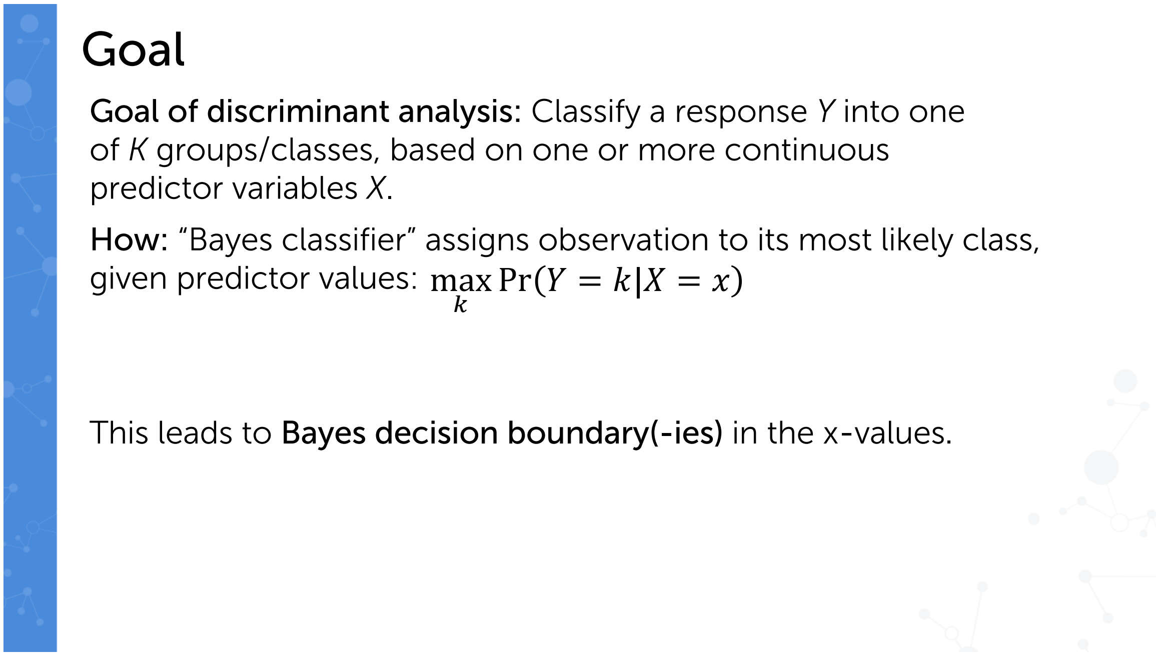
Theory of LDA

Important note: Transcripts are **not** substitutes for textbook assignments.

Learning Objectives

By the end of this lesson, you will be able to:

- Understand basic development of LDA classification.
- State underlying assumptions for LDA classification with a single predictor.
- Recognize situations appropriate for single-predictor classification.
- Understand how assumptions enter into LDA classification.
- Apply single-predictor LDA using *lda* function and check assumptions.
- Assess predictive abilities via cross-validation.



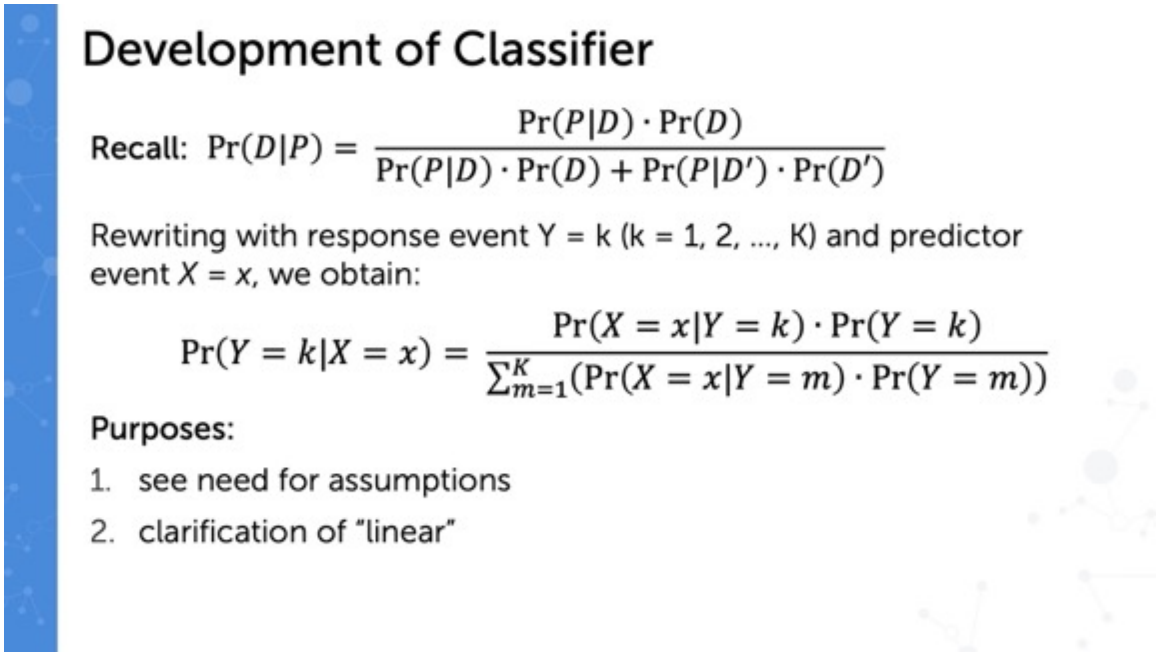
Goal

Goal of discriminant analysis: Classify a response *Y* into one of *K* groups/classes, based on one or more continuous predictor variables *X*.

How: “Bayes classifier” assigns observation to its most likely class, given predictor values: $\max_k \Pr(Y = k | X = x)$

This leads to **Bayes decision boundary(-ies)** in the x-values.

Recalling the goal of discriminant analysis. In this lecture, we will explore the connection from a single predictor used through Bayes' rule and certain assumptions to classify a qualitative response y into K classes. To do this, we'll use the so-called Bayes' classifier, which looks at the class whose conditional probability, given the predictor is highest. This further allows us to define intervals in the x space, in which each class is the prediction from the classifier. These intervals are marked apart by what are known as Bayes' decision boundaries.



Development of Classifier

Recall: $\Pr(D|P) = \dfrac{\Pr(P|D)\cdot\Pr(D)}{\Pr(P|D)\cdot\Pr(D)+\Pr(P|D')\cdot\Pr(D')}$

Rewriting with response event Y = k (k = 1, 2, ..., K) and predictor event $X = x$, we obtain:

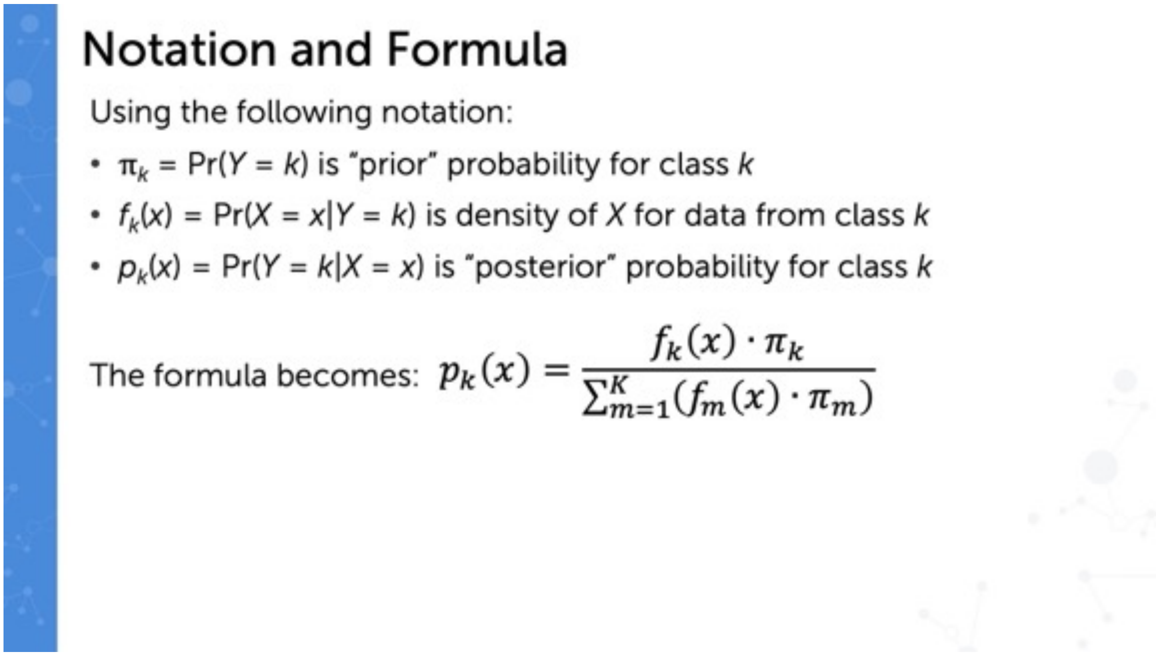
$$\Pr(Y=k|X=x) = \frac{\Pr(X=x|Y=k)\cdot\Pr(Y=k)}{\sum_{m=1}^{K}(\Pr(X=x|Y=m)\cdot\Pr(Y=m))}$$

Purposes:

1. see need for assumptions
2. clarification of "linear"

We begin with Bayes' rule as expressed in the previous example framework. The outcome disease is represented with a d and has two possibilities-- has or has not-- and test outcome is represented as a p for positive or an n for negative. We rewrite this more generically. Rather than the response having necessarily two possible outcomes, we let the response-- variable y-- be one of capital K possible classes.

Then for some continuous variable x, the conditional probability of y being in class K given x can be written as in the equation displayed, simply by substituting in the response values for the outcome of-- or the observed disease value and substituting the x values in place of the positive or negative. While this starts to get messy walking through the development serves two purposes. First, we can observe the need to have certain assumptions based on the process. Second, clarification of the word linear in the method name is found from the form of the classifier decision boundaries.



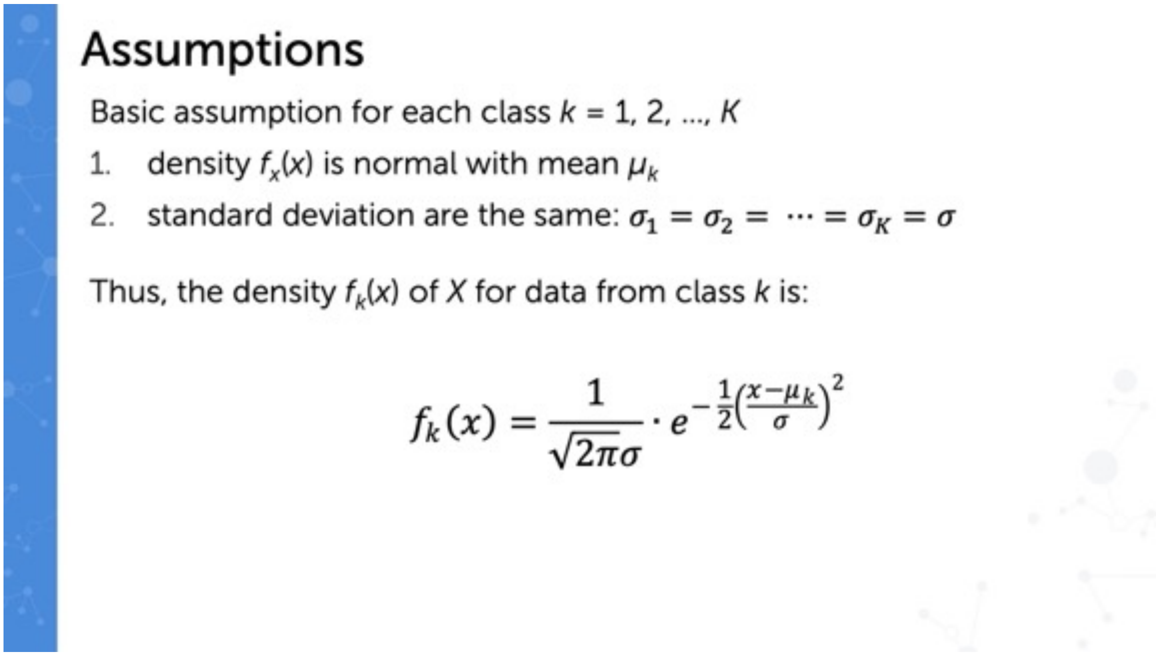
Notation and Formula

Using the following notation:

- $\pi_k = \Pr(Y = k)$ is "prior" probability for class k
- $f_k(x) = \Pr(X = x|Y = k)$ is density of X for data from class k
- $p_k(x) = \Pr(Y = k|X = x)$ is "posterior" probability for class k

The formula becomes: $p_k(x) = \dfrac{f_k(x)\cdot\pi_k}{\sum_{m=1}^{K}(f_m(x)\cdot\pi_m)}$

Using notation that better represents the form of the inputs, we use pi sub K for the probability-- the unconditional probability-- of the outcome y. The response y being in class K. We let f sub K evaluated at x represent a density for the continuous x. That is, the probability density function of x at a given value of y equals K. And Similarly, we let p sub K of x represent the probability that y equals K given x equals value little x. Substituting these in for the original probability notation, we get the form from the book that shows p sub K of x as a function of the f sub K and pi sub K.



Assumptions

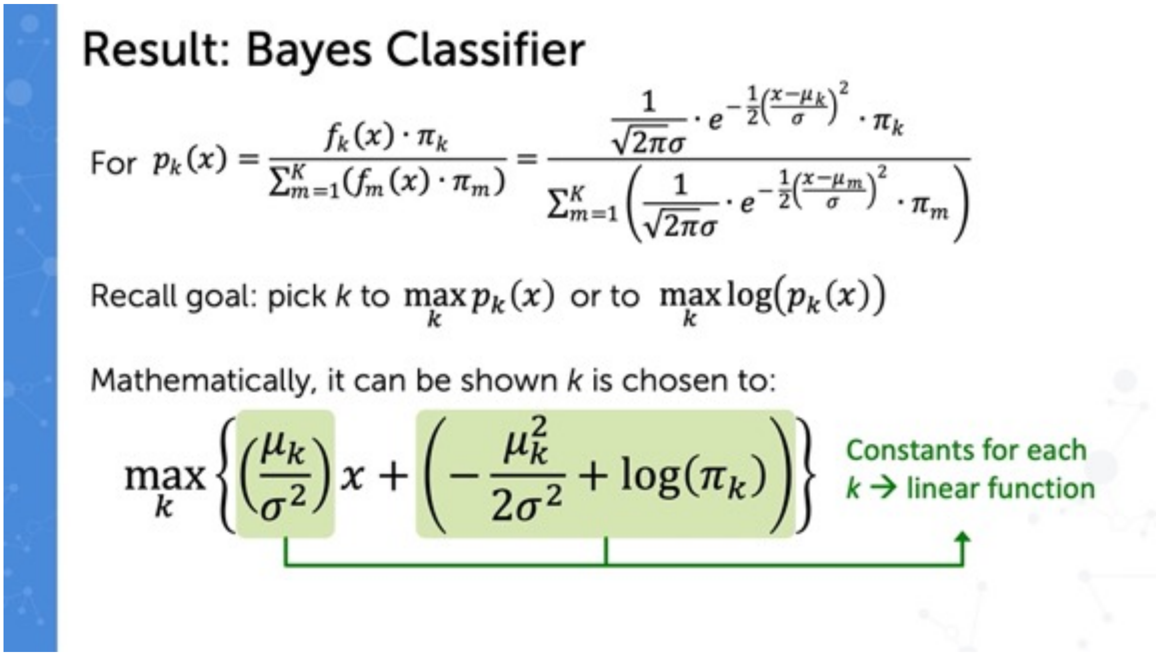
Basic assumption for each class k = 1, 2, ..., K

1. density $f_x(x)$ is normal with mean μ_k
2. standard deviation are the same: $\sigma_1 = \sigma_2 = \cdots = \sigma_K = \sigma$

Thus, the density $f_k(x)$ of X for data from class k is:

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}$$

The rather typical assumption of a normal distribution for the continuous variable is employed here. That is, the density f sub K is identified to be normal. And for each class K may have a different mean, mu sub K. Initial derivation will additionally assume all classes have the same standard deviation sigma. The normal density function can then be written out and substituted into the previous formula.



Result: Bayes Classifier

For $p_k(x) = \frac{f_k(x)\cdot\pi_k}{\sum_{m=1}^{K}(f_m(x)\cdot\pi_m)} = \frac{\frac{1}{\sqrt{2\pi\sigma}}\cdot e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}\cdot\pi_k}{\sum_{m=1}^{K}\left(\frac{1}{\sqrt{2\pi\sigma}}\cdot e^{-\frac{1}{2}\left(\frac{x-\mu_m}{\sigma}\right)^2}\cdot\pi_m\right)}$

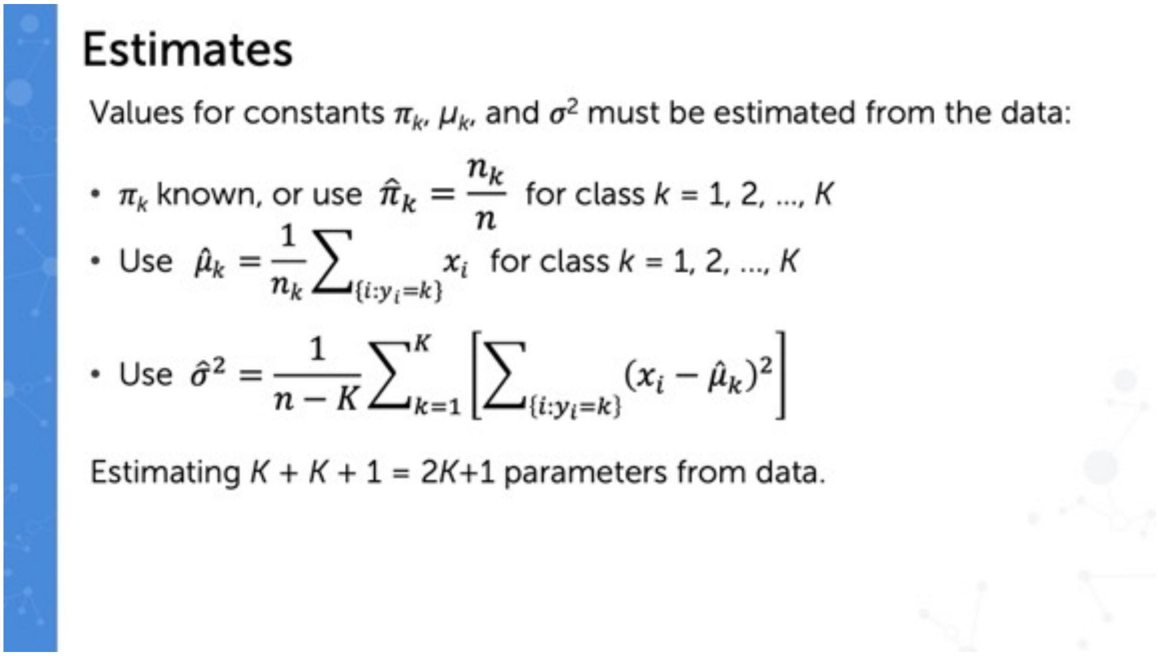
Recall goal: pick k to $\max_k p_k(x)$ or to $\max_k \log(p_k(x))$

Mathematically, it can be shown k is chosen to:

$$\max_k\left\{\left(\frac{\mu_k}{\sigma^2}\right)x + \left(-\frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)\right)\right\}$$

Constants for each k → linear function

As a result, we get a final workable form of p sub K of x, the conditional probability that y is in class K given x. While it is messy to maximize directly, we can equivalently maximize the log of p sub K of x. And this works out to a simpler form. That is, maximize over all K some constant times x plus another constant, where those constants involve the mu sub K and pi sub K pieces that vary with K, and sigma squared, which is constant for all K. Now, note that this is a linear function in x, which is very simple to maximize and provides the reason for the name of the method linear discriminant analysis.



Estimates

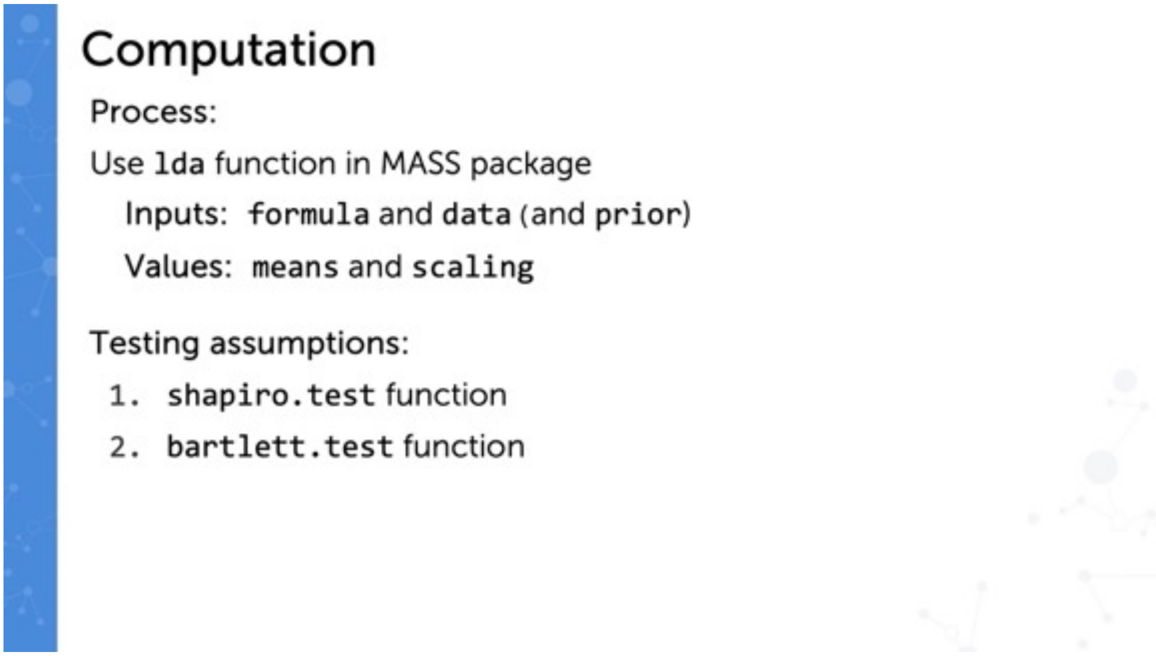
Values for constants π_k, μ_k, and σ^2 must be estimated from the data:

- π_k known, or use $\hat{\pi}_k = \frac{n_k}{n}$ for class $k = 1, 2, \ldots, K$
- Use $\hat{\mu}_k = \frac{1}{n_k} \sum_{\{i: y_i = k\}} x_i$ for class $k = 1, 2, \ldots, K$
- Use $\hat{\sigma}^2 = \frac{1}{n - K} \sum_{k=1}^{K} \left[\sum_{\{i: y_i = k\}} (x_i - \hat{\mu}_k)^2 \right]$

Estimating $K + K + 1 = 2K+1$ parameters from data.

So we have constants, but we don't know their values. Hence, we must estimate them from sample information. Occasionally, we might have some prior information about the pi sub Ks. But typically, we will have to estimate the class proportion pi hat sub K as simply the proportion of each class in the sample. And we will come up with K estimators.

Additionally, we will get K more estimators in order to come up with an estimate for each class mean, which is simply calculated as the sample average of the x sub i's in that class. The overall variance is estimated with a typical sample variance formula. But using means corresponding to each x sub i's class and dividing by n minus K to obtain an unbiased estimator. In total, we are estimating K plus K plus 1 parameters.



Computation

Process:

Use `lda` function in MASS package

Inputs: `formula` and `data` (and `prior`)

Values: `means` and `scaling`

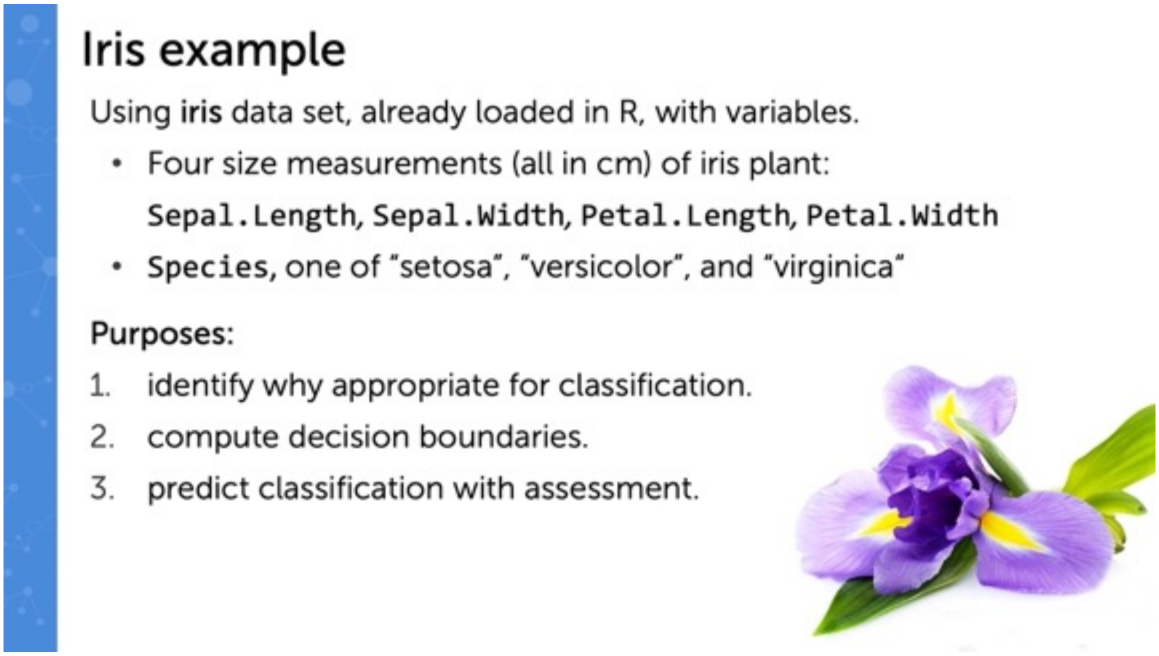
Testing assumptions:

1. `shapiro.test` function
2. `bartlett.test` function

While such computation can be done by hand, it is painstaking. Happily, there is a very user friendly function in the mass package called simply lda, all in lowercase letters. Typically, only two inputs will be needed. Formula is the predictive model to be used and data references the appropriate set of information. We will be using the shapiro.test function to test the assumption of approximate normality of data and the bartlett.test function to test the assumption of equal standard deviations.

Notes:

R Manual Pages: lda function, shapiro.test function, bartlett.test function.



Iris example

Using **iris** data set, already loaded in R, with variables.

- Four size measurements (all in cm) of iris plant:
  `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`
- `Species`, one of "setosa", "versicolor", and "virginica"

Purposes:

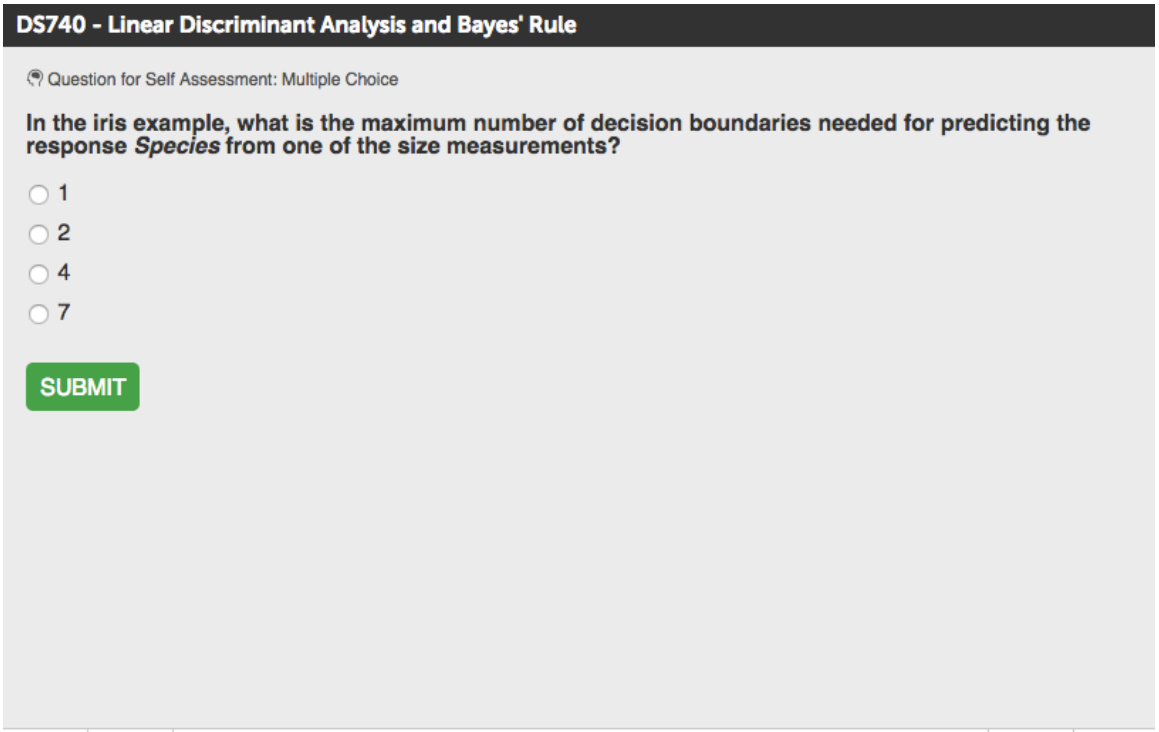
1. identify why appropriate for classification.
2. compute decision boundaries.
3. predict classification with assessment.

A very famous historical data set is the Iris dataset. It was analyzed by Fischer all the way back in 1936 and is used to illustrate LDA, or Linear Discriminant Analysis. The Iris data includes 50 flowers from each of the three species and four size measurements-- two of the petal and two of the sepal-- of the plants. We will visually explore classification and run the lda function to get decision boundaries and predict classification in our video in R.

Notes:

Link: Iris data set

Question 1



DS740 - Linear Discriminant Analysis and Bayes' Rule

Question for Self Assessment: Multiple Choice

In the iris example, what is the maximum number of decision boundaries needed for predicting the response *Species* from one of the size measurements?

○ 1

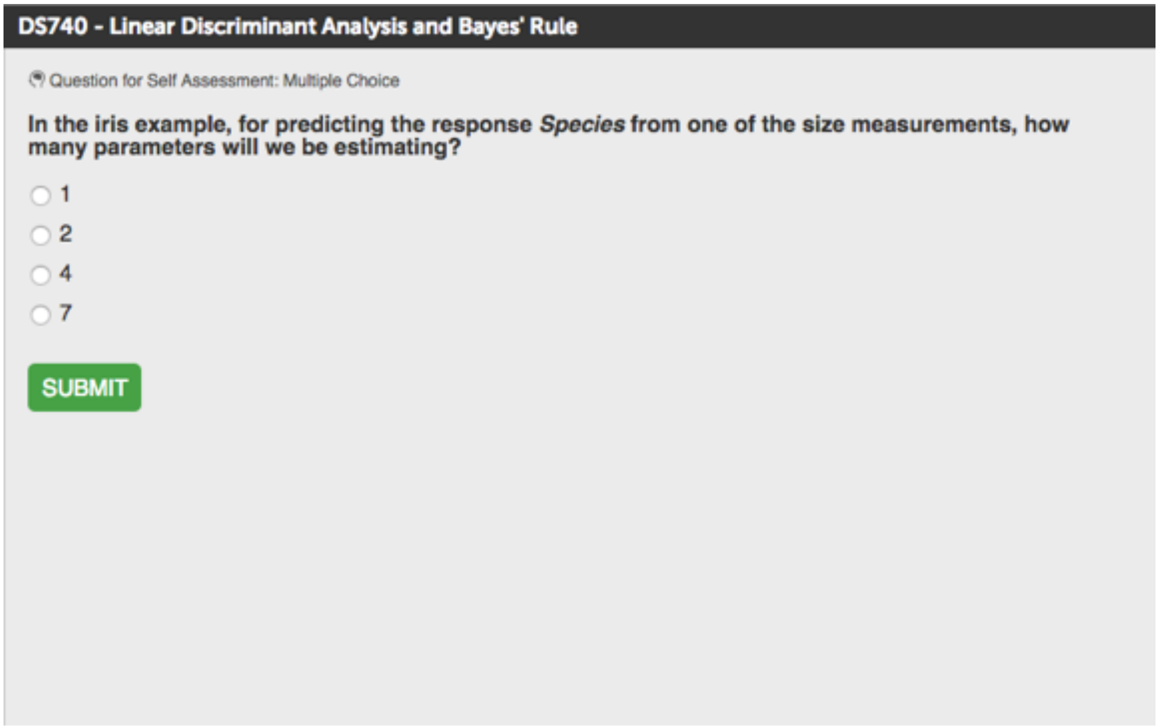
○ 2

○ 4

○ 7

Answer is at the end of this transcript

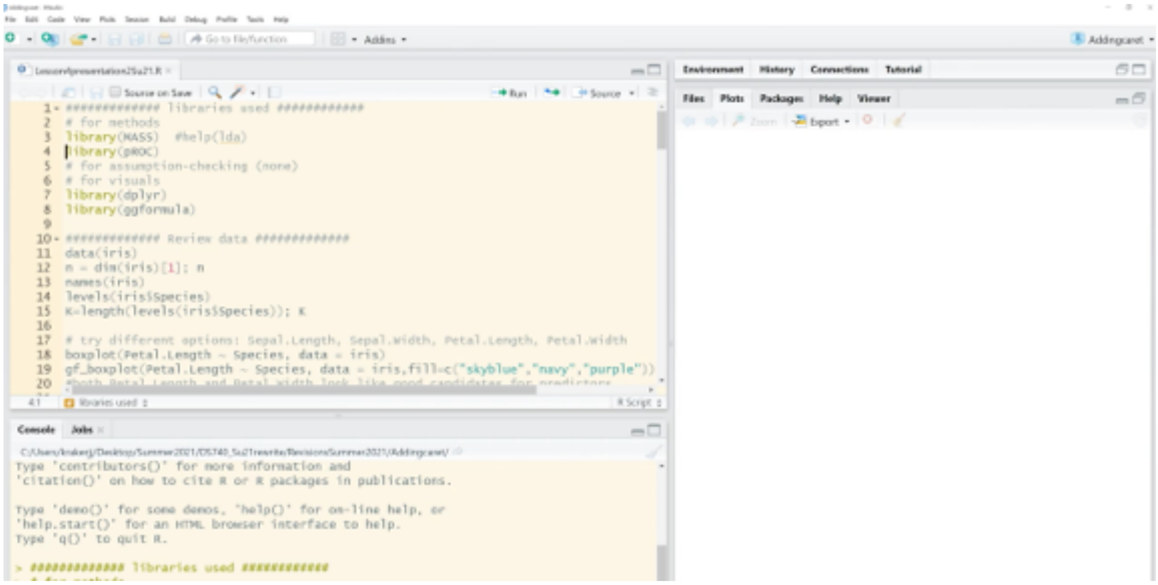
Question 2


DS740 - Linear Discriminant Analysis and Bayes' Rule
Question for Self Assessment: Multiple Choice

In the iris example, for predicting the response *Species* from one of the size measurements, how many parameters will we be estimating?

○ 1
○ 2
○ 4
○ 7

Answer is at the end of this transcript


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Lessonfpresentation2Su21.R
Source on Save
Run
Source
Environment History Connections Tutorial
Files Plots Packages Help Viewer
Zoom
Export
libraries used
R Script
Console
Jobs

```
############# libraries used ############
# for methods
library(MASS)  #help(lda)
library(pROC)
# for assumption-checking (none)
# for visuals
library(dplyr)
library(ggformula)

############# Review data #############
data(iris)
n = dim(iris)[1]; n
names(iris)
levels(iris$Species)
K=length(levels(iris$Species)); K

# try different options: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
boxplot(Petal.Length ~ Species, data = iris)
gf_boxplot(Petal.Length ~ Species, data = iris,fill=c("skyblue","navy","purple"))
#both Petal.Length and Petal.Width look like good candidates for predictors
```

```
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> ############# libraries used ############
```

This slide represents a video/screencast in the lecture. The transcript does not substitute video content.

Today, we're going to be working with the LDA function, which is contained in the MASS library. And since we'll also be working with a ROC function example, we'll open up the library for working with ROC functions. We don't have any specialty or additional libraries for assumption checking. And for a couple of the visuals, we will work with ggformula and dplyr commands.

We're going to be taking a look at the iris data today, which is already included in the usual Base R. So we'll begin by, if you have not seen this data set before, taking a look at the size, as well as the number of levels of the categorical variable that we'll be using as the response species. Species has three levels-- setosa, versicolor, and virginica-- which are the species of iris plants.

We could take a look at a Base R visual of one of the variables. Here, I chose petal length, but you could try some of the other numeric predictor variables. So if we take a look at iris, the data set itself, we see that our categorical response variable will be species. And we have four potential predictor variables. We are currently looking at petal length, which is the most discerning or discriminating.

Another visual that we could use is a ggformula function called gf boxplot. That just makes it a little bit easier to fill in colors, for example. And so I've picked out three colors to denote the levels of species.

And if you take a look at the values of petal length, setosa, obviously, is very far separated in terms of the petal lengths. Versicolor and virginica have some overlap. And so one question might be, how well can we actually distinguish one of those two, let's say, virginica? Thus, I might define or might be working with an indicator type response. And in this case, if I take a look at virginica, it's a 01 indicator of the virginica species.

And so for our very first application, in terms of applying the LDA function fit, we might wish to just work with that, either virginica or not type of response. And part of the reason for this is I'd like to illustrate an interesting characteristic of the ROC functions when working with LDA. So if I first include a ROC curve with the response as this 01 indicator value, and my predictor as simply the numeric values of petal length. And I make a ROC curve and plot it. When I get the same sort of values from the posterior probabilities of an LDA fit, I'm going to wind up with exactly the same curves.

So let's talk through this. For my iris data, I model this 01 virginica response on petal length using linear discriminant analysis LDA function. I store that. And then when I take a look at the predictions of that model fit using my iris data, if I just run that code, I get quite a bit of output. And so sifting through this is really quite important.

The most important piece that we want from this is the posterior probability. And its the posterior probability for the level zero and one, where one indicates the virginica species. So it's this second column, the posterior probabilities for level one that I want. So I'll take a look at just the posterior probabilities and pull off the second column of those.

When I do that, and store those posterior probabilities as ldaprob, using those as my predictor values is going to give me exactly the same curve because of the linear relationship with the predictor values. So if you kept your eye on the plot, it didn't change at all. So that's just an example of linear discriminant analysis with a two class categorical response variable.

We actually want to apply LDA, linear discriminant analysis, with the three class species categorical variable. Not much to adjust here, specifically if we run the LDA function on the model only, now using species with the three categories setosa, versicolor, and virginica as the response, and petal length is the predictor, we'll still get a fit similar to what we saw before. And if we made a prediction, that is we re-predict using LDA fit the same data used to fit that model, we would get the posterior probabilities as we saw before. But now, posterior probabilities for setosa, versicolor, and virginica classes.

OK, so I would like to figure out, because as well as part of this prediction, I get the classification output. So just taking a look at the class as predicted on the data used to fit the model, I'm going to store that in cred class and compare that to my original response, which I'm going to label as y. Making a table, we see that there's a very accurate method here, or at least it's very accurate when applied to the data used to fit the model. Two of the actual versicolors are misclassified as virginicas. And six of the actual virginicas are misclassified as vericolors, as we can see from this table, where the y, the actual observed response, is in the rows, and the predictions are the columns.

If we summarize this, we get an error of only about 0.0533. Now, we're not comparing this to any other models at this time, but that's a pretty low error rate. We should make sure that we're getting an honest idea of our error rate, though. And so we're going to use cross-validation to proceed with this.

We'll set up our storage locations for the predictions. We'll set up our CV groups, as we've done in prior lessons. And then we'll loop through those CV groups consecutively designating each group or setting up a logical vector to indicate each group, getting the train set, the test set. And then fitting the LDA models previously specified only to the training set and predicting the outcome.

Now, as with any process, I highly recommend going through the steps inside the loop before you try to run the loop to make sure it works. So what I've done is I start by setting ii equals one, and running to make sure I'm designating the right things with each line. If I wasn't sure what was showing up in group ii, that it's a logical vector, I could check. Does seem to be a logical vector.

I define my train and test set. Might be nice to see that those are of the approximate correct sizes, 90% and 10% of the data, respectively. All that looks good.

I store, again inside the loop, eventually LDA fit ii. And I might just want to take a look at is that actually appear to have the characteristics of an LDA fit? Yeah, seems to work well. Interestingly, now the prior probabilities of the groups, because we have a random train set, they're not all quite evenly split, but pretty close to evenly split in the train set.

It is possible to predict and simply store the predicted values for the test set in their class four. But to make this compatible with later visuals, we'll actually turn these predicted values into the text with the AS character function. So if I run that section, it's just the same values, but as text. And store those as my predictions.

And then, finally, put those predictions in the correct locations on my full storage vector for predictions. So if I take a look at the CV pred class, it should contain about 10% of the locations filled in with the corresponding predicted text. Well, it looked like everything ran correctly. So let me run this through my CV groups.

And then let's tabulate. So I'm going to table my previously defined labeled y, my response values, my y versus the cross-validated predicted class. And here, we actually still get just a 5% error rate. But now, there are three actual versicolors that are misclassified as virginica and five actual virginicas that are misclassified as versicolor. So the mix is a little bit different when we do this honest prediction, but the misclassification rate is right about where we saw it previously when we reclassified the data used to fit the model.

The last step here is to see if it's reasonable to use LDA. And we know that one of the assumptions that we're working with in LDA is that we can use a common variance for the groups, as well as have normality of the predictor values. When taking a look at the petal length as my xvar values, I'm going to subset using the brackets only where our species is setosa of the xvar and store that as xSetosa. And do the same sort of thing for versicolor and virginica.

Running normality tests on the setosa petal lengths, it's close to concerning, but that's because of a slightly more discreet nature of the values. With versicolor, it's very reasonable. Normality is reasonable. There's no strong evidence against it. And same thing with virginica. So normality looks great.

However, if I run a test about looking for evidence of non-constant variance, that is highly, highly significant with a very, very small p-value. And piping the iris data set grouping by species and then summarizing would be a way to take a look at those standard deviations. Equivalently, you could simply take the standard deviation of xSetosa and of xVersicolor and of xVirginica and find the standard deviations. So we can see that setosa's standard deviation is much smaller, and thus, equal variances are not reasonable. We really want to use QD, and we'll talk about that in the next presentation.

That concludes the application of LDA. As a bonus, I have included some visualizations of the boundaries. I won't go through the coding in detail other than to note that I am computing a variety of statistics among the different groups, the different classes of species, and then visualizing the linear decision boundaries.

So if I were to make a histogram of the values in each of those three groups-- setosa, versicolor, and virginica-- I also plotted a linear decision boundary that we look to maximize, so up to a petal length of 2.861. That is about right here. The highest in the vertical axis value of this linear bound is for setosa. Between

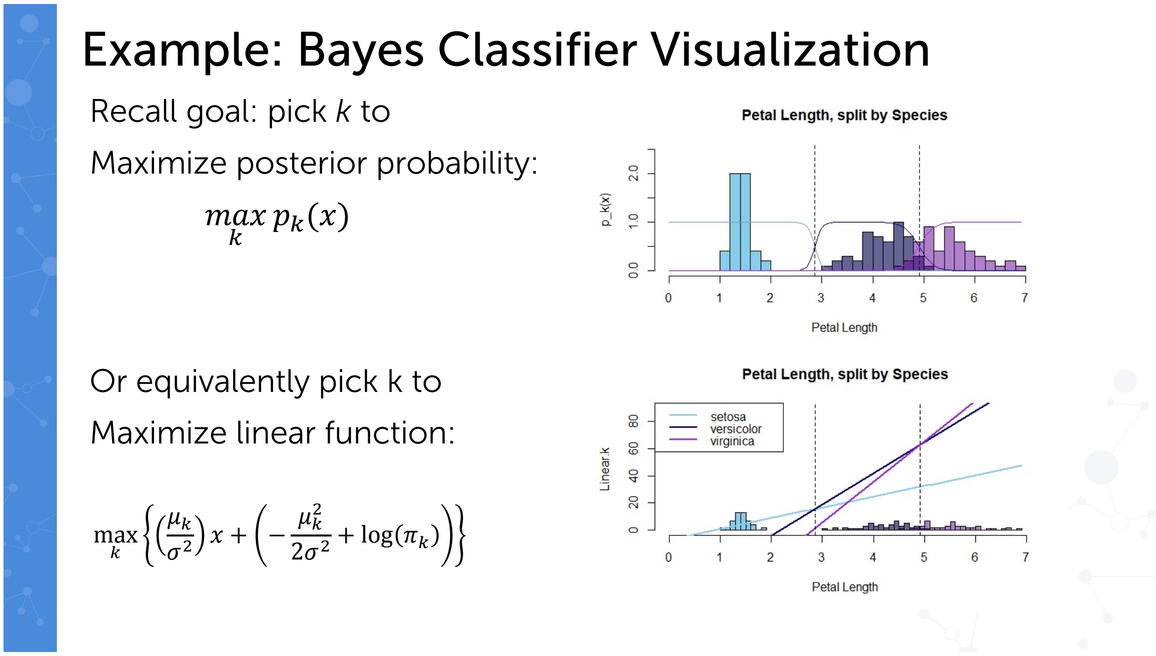
2.861 and 4.906. The highest line is that for versicolor. And finally, for petal lengths above 4.906, we would classify as virginica because the line for virginica is higher after that point.

The next visual is an attempt to correspond this to our original goal functions, that is the posterior probabilities that you saw, both as part of the output and which had a computation on a prior slide. This is a little bit more complex, but I believe more visually understandable in terms of what we're actually looking to do. So let's take a look at this plot, which shows the corresponding cut points. But now, what you see is the posterior probabilities.

And this makes more sense, at least in an understanding in a probabilistic sense, because we're talking about a posterior probability very close to one. And then it suddenly drops off for setosa. And suddenly, the posterior probability for versicolor is higher and close to one and then drops off a little bit. And the reason it's dropping off less sharply, more gradually, is because there is some overlap between the versicolor and virginica species. And after the value 4.906, the posterior probability for virginica is highest.

The visuals are also summarized and displayed in the next slide. I hope that that helps explain some of the correspondence between maximizing the linear function with the maximization of the posterior probability.
Notes:

See the online course for a downloadable R file containing the set of commands used in this demonstration.



Example: Bayes Classifier Visualization

Recall goal: pick *k* to

Maximize posterior probability:

$$\underset{k}{max}\, p_k(x)$$

Or equivalently pick k to

Maximize linear function:

$$\max_k\left\{\left(\frac{\mu_k}{\sigma^2}\right)x+\left(-\frac{\mu_k^2}{2\sigma^2}+\log(\pi_k)\right)\right\}$$

Here, we show the plots demonstrated in the recording on the previous slide, with the top plot showing where the posterior probability values are maximized and identifying the cut points, and then showing that the linear functions are maximized with cut points occurring at the same locations. This explains why the method is called "linear discriminant analysis"-- that is, because the functions that are maximized are simply thought of and worked with as linear functions.

Summary One

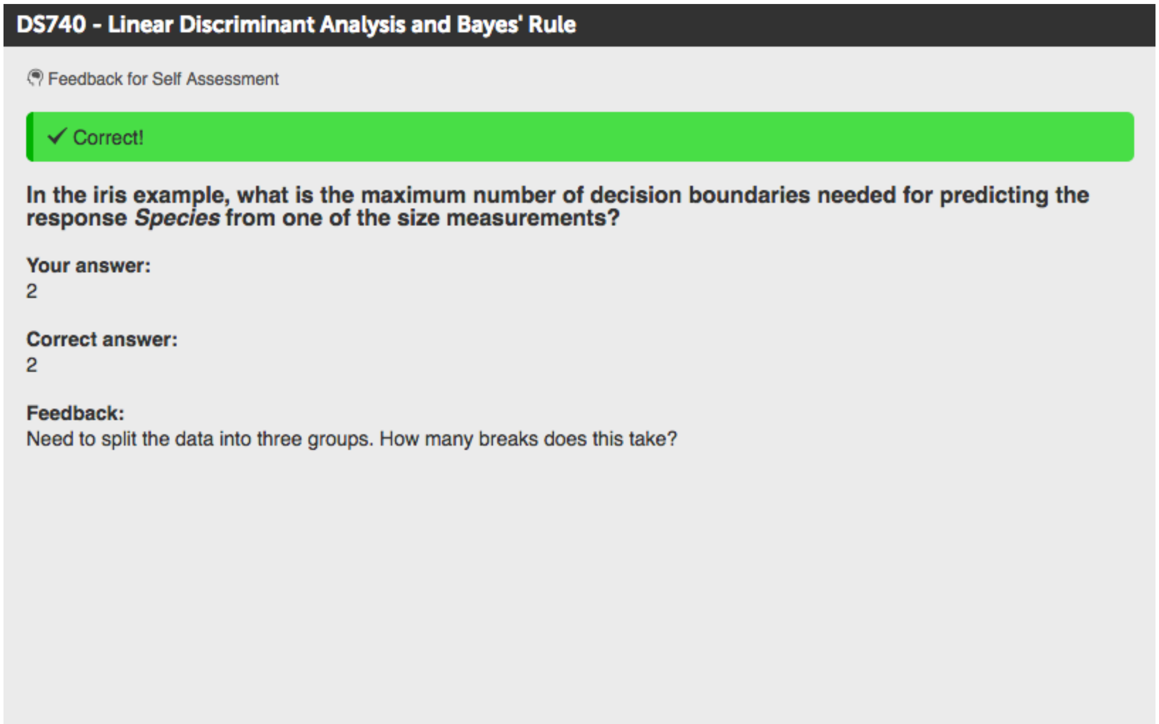
We walked through development of the goal function of LDA with one predictor variable.

- This goal function comes from the foundation of Bayes' rule, and uses notation appropriate to the values and forms that the response Y and predictor X can take.
- It is maximized to identify the value of the Bayes' classifier.

Summary Two

- Assumptions about the form (normal) of the density of *X* and the (same) standard deviations of the *x*-values within each class allow a simpler goal function to be derived → a linear function of *x*
- Sample data is used to estimate the constants in the goal function, producing decision boundaries and predicted classes through use of the `lda` function.

Question 1 Answer



DS740 - Linear Discriminant Analysis and Bayes' Rule

Feedback for Self Assessment

✓ Correct!

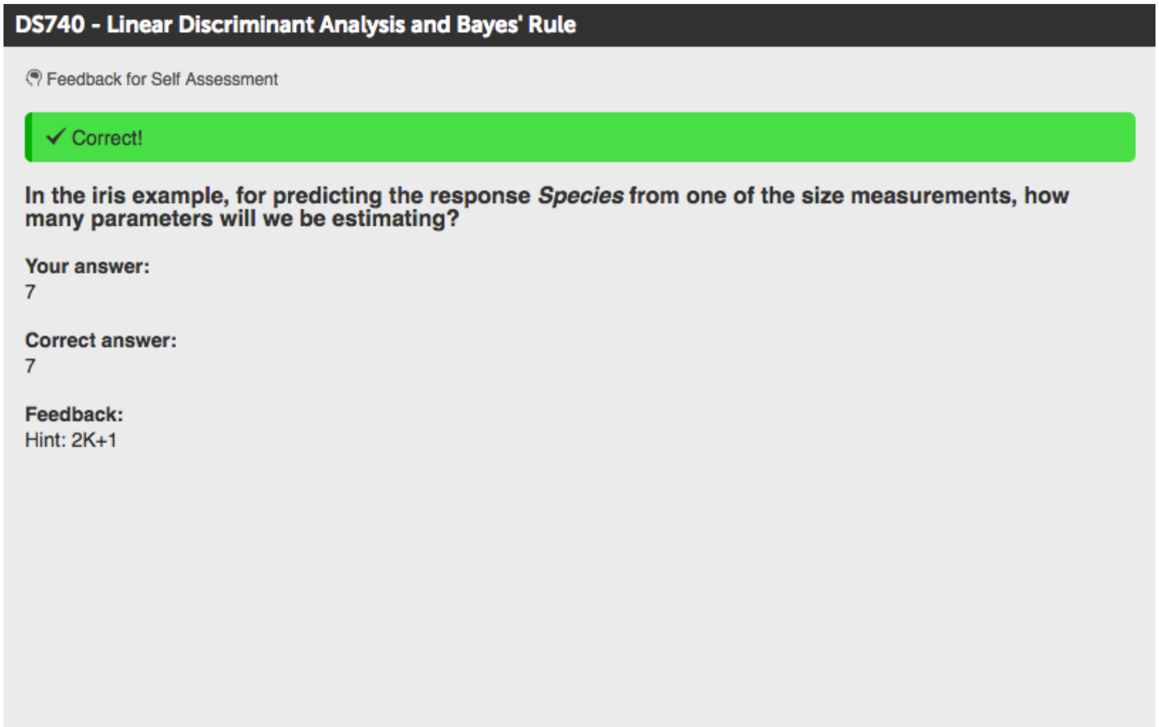
In the iris example, what is the maximum number of decision boundaries needed for predicting the response *Species* from one of the size measurements?

Your answer:
2

Correct answer:
2

Feedback:
Need to split the data into three groups. How many breaks does this take?

Question 2 Answer



DS740 - Linear Discriminant Analysis and Bayes' Rule

Feedback for Self Assessment

✓ Correct!

In the iris example, for predicting the response *Species* from one of the size measurements, how many parameters will we be estimating?

Your answer:
7

Correct answer:
7

Feedback:
Hint: 2K+1