**Discriminant Analysis (DA)**

**What is DA?**

Suppose N observations are known to come from K different populations (or groups). Each observation’s group is known. DA allows for the construction of a mathematical “rule” to classify observations into the populations. When data on new observations are collected (without knowing the population of the observed data), this rule can be applied to classify the observations.

Example: Placekicking data

Predict which placekicks will be a success or a failure using variables such as distance, wind, … .

Example: Wheat kernels

A wheat researcher wanted to classify wheat kernels into “healthy”, “sprout”, or “scab” types. He developed an automated method to take the following measurements: kernel density, hardness index, size index, weight index, and moisture content. DA in this situation allows one to determine if the wheat kernels could be accurately classified into the three kernel types using the five measurements.

Example: Artillery shells

The army was interested in developing ways to determine the contents of unexploded artillery shells without exploding or contacting them.



DA was used to help differentiate between 155mm artillery shells that were empty, sand filled, …, mustard gas filled.

DA is somewhat similar to regression analysis because both are using independent (explanatory) variables to predict a dependent (response) variable. In regression analysis, the dependent variable is quantitative. In discriminant analysis, the dependent variable is qualitative (categorical). Another important difference between the two is that in DA there is must less emphasis on interpreting the effect that the independent variable has on the dependent variable. The top priority is to obtain the highest classification accuracy possible.

A whole course could be taught on DA and related classification methods!

**Two populations**

Suppose there are two multivariate normal populations denoted by Π1 and Π2. Therefore, Π1 corresponds to Np(μ1, Σ1) and Π2 corresponds to Np(μ2, Σ2).

Let **x** denote an observation from one of these populations. The goal of discriminant analysis is to be able to predict the population of **x**.

Four different discrimination rules

# Likelihood rule

The likelihood function here is just the multivariate normal probability density evaluated at **x**:





Choose Π1 if L(μ1, Σ1| **x**) > L(μ2, Σ2| **x**) and choose Π2 otherwise.

Why should this be used as a rule?

# Linear discriminant rule

# Suppose Σ1 = Σ2. The likelihood rule simplifies to:

Choose Π1 if **b**′**x** – k > 0 and choose Π2 otherwise where **b** = **Σ**-1(**μ**1-**μ**2) and k = (1/2)(**μ**1-**μ2**)′Σ-1(**μ**1+**μ2**).

**b**′**x** is called the *linear discriminant function* of **x**. It is named this because the linear combination of **x** summarizes all of the possible information in **x** that is available for discriminating between the two populations.

# Mahalanobis distance rule

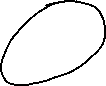
Suppose Σ1 = Σ2. Let

di = (**x** – **μ**i)′**Σ**-1(**x** – **μ**i)

for i = 1, 2, be the Mahalanobis distance between **x** and the mean for population i. Compare this to the Euclidean distance measure.

Choose Π1 if d1 < d2 and choose Π2 otherwise.

To help explain this rule, consider the plot below for a bivariate situation:



The two ellipses on the plot are the same contours for both normal populations (contours are meant to be the same size). Suppose an observation **x**0 is observed. The distance **x**0 is from **μ**1 and **μ**2 is calculated taking into account the covariance matrix **Σ**. The smaller distance corresponds to which population **x**0 will be classified into.

# Posterior probability rule

Suppose **Σ**1 = **Σ**2. The *posterior probability* of **x** being in population i is





Note that this is not really a “probability” because there is not a random event being considered (**x** is either from population 1 or 2). This just gives a number between 0 and 1 to measure the confidence of being in population i.

For example, a posterior probability near 1 means there is a lot of confidence that **x** comes from population i. A posterior probability near 0 means there is little confidence that **x** comes from population i. With two populations, a posterior probability near 0.5 indicates indecision between the two populations.

Choose Π1 if P(Π1|**x**) > P(Π2|**x**) and choose Π2 otherwise.

Because **μ**i and **Σ**i are usually never known, their corresponding estimates replace the parameters in the above four rules. When there is belief that Σ1 = Σ2, the “pooled” estimate of the covariance matrix is



where Ni is the sample size from population i. This pooled estimate is very similar to the pooled estimate of variance typically taught when performing a hypothesis test for the difference between two means. The difference now is that we are using matrices here instead of single variances for each population.

Estimating probabilities of misclassification

We need to determine the accuracy of a discriminate rule. This is done by examining the percentage of correct and incorrect classifications in one of four ways.

# Collecting new data

This is probably the best method in terms of getting good estimates of the discrimination rule’s accuracy.

The discriminant rule is found on the original data set. Then a new data set (with the populations known) is collected to try out the discrimination rule. The percentage of correct and incorrect classifications on this new data set measures the rule’s accuracy.

What is the problem with this method?

# Estimates from holdout data

Find a discriminant rule on part of the data set and try it out on the other part. This is done by first randomly removing a portion of your observations from the data set and putting this data aside. With the remaining part of the data, find the discriminant rule. Now, try the rule out on the data put aside just like you would if a whole new data set was collected as described in 1. This is equivalent to having a model building (training) and a validation (calibration, test) data set in regression analysis.

What is the problem with this method?

# Resubstitution

Classify the data in which the discriminant rule was formed upon. What is the problem with this method?

# Cross-validation estimates

Remove the first observation from the data set and find a discriminant rule using the remaining N – 1 observations. Predict the classification of the first observation. Put the first observation back into the data set and remove the second. Find a discriminant rule on the remaining N – 1 observations. Repeat this process for each observation. The resulting N different classifications can be used to find a nearly unbiased estimate of the discrimination rule’s accuracy. This “leave-one out” method is a form of jackknifing.

What is the problem with this method?

If given the choice between resubstitution or cross-validation, use cross-validation because it generally provides results that are less biased. Consequently, the percent of misclassifications are generally higher for cross-validation than for resubstitution.

Example: Placekicking data (PlacekickDA.R, Placekick.csv, valid.csv)

Please see the description of the data from earlier in the course. Our goal here is to develop a discriminant rule that classifies each observation as a success or failure. We will use methods 2, 3, and 4 to examine how well this rule works. The model building data is Placekick.csv (about 83% of the observations), and the validation data set is valid.csv (about 17% of the observations).

With respect to this data, what do you think about the multivariate normal distribution assumption that DA requires?

DA often still works decent even if the assumptions are violated. The most important aspect of a DA is the accuracy. If the accuracy is acceptable (as evaluated by cross-validation, new data, or validation data), then use it!

The main R function used for DA is lda(), and it is in the MASS package. The reason for the “l” in the name is that we are using a specific type of DA known as “linear” discriminant analysis. We will discuss where this name comes from later.

Reading in the data:

> placekick <- read.csv(file = "Placekick.csv")

> head(placekick)

week distance change elap30 PAT type field wind good

1 1 21 1 24.7167 0 1 1 0 1

2 1 21 0 15.8500 0 1 1 0 1

3 1 20 0 0.4500 1 1 1 0 1

4 1 28 0 13.5500 0 1 1 0 1

5 1 20 0 21.8667 1 0 0 0 1

6 1 25 0 17.6833 0 0 0 0 1

> valid <- read.csv(file = "valid.csv")

> head(valid)

week distance change elap30 PAT type field wind good

1 1 21 0 15.8500 0 0 1 0 1

2 1 47 1 0.0000 0 1 0 0 1

3 1 51 1 19.7833 0 0 1 0 0

4 1 20 0 22.1500 1 0 0 0 1

5 1 20 0 4.0167 1 0 0 0 1

6 1 20 0 23.7167 1 0 1 0 1

Plots of the data can be performed to obtain an initial understanding of it. Note that the parallel coordinates plot is not too helpful here due to the large number of discrete variables in the data set. Using similar code as we have seen before, below are plots obtained from using PCA using all of the variables except for good:

> pca.cor <- princomp(formula = ~ week + distance + change + elap30 + PAT + type + field + wind, data = placekick,

cor = TRUE, scores = TRUE)

> summary(pca.cor, loadings = TRUE, cutoff = 0.0) #44% with 2PCs, 58% with 3PCs

Importance of components:

Comp.1 Comp.2 Comp.3

Standard deviation 1.3904744 1.2585231 1.0556628

Proportion of Variance 0.2416774 0.1979850 0.1393030

Cumulative Proportion 0.2416774 0.4396624 0.5789654

Comp.4 Comp.5 Comp.6

Standard deviation 1.0144812 0.9663143 0.9107749

Proportion of Variance 0.1286465 0.1167204 0.1036889

Cumulative Proportion 0.7076120 0.8243324 0.9280212

Comp.7 Comp.8

Standard deviation 0.58988314 0.47735534

Proportion of Variance 0.04349526 0.02848351

Cumulative Proportion 0.97151649 1.00000000

Loadings:

Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7

week 0.022 0.049 0.579 0.278 0.756 0.084 0.070

distance 0.644 -0.079 -0.016 0.065 0.023 -0.295 -0.011

change 0.332 -0.125 -0.054 -0.433 0.106 0.817 0.033

elap30 -0.138 0.043 0.008 -0.825 0.348 -0.419 -0.027

PAT -0.649 0.117 0.017 0.007 -0.015 0.237 -0.014

type -0.132 -0.699 0.127 -0.032 -0.090 -0.080 0.680

field -0.122 -0.681 -0.173 0.099 0.206 -0.014 -0.662

wind 0.030 -0.086 0.784 -0.198 -0.495 -0.002 -0.304

Comp.8

week 0.015

distance -0.698

change -0.061

elap30 -0.021

PAT -0.713

type -0.030

field 0.001

wind -0.009

> #Plotting code has been excluded here

A graph with red dots

Description automatically generated

A diagram of a graph

Description automatically generated

Two PCs explain only 44% of the total variation in the data, and three PCs explain only 58% of the total variation in the data. Also, the larger the distance, the smaller PC #1 is. Because the longer a placekick is the less likely it is to be successful, there are more black points on the right side of the PC score plot.

The bands in the plot are due to the discreteness in the data. For example, below are two plots where the points are colored corresponding to PAT and field, respectively:

A graph with green and blue dots

Description automatically generated

A graph of a graph with green and blue dots

Description automatically generated

Overall, we obtain here a general sense for what types of placekicks may be successful, but there is not one obvious way that the observations can be categorized as a success or failure.

Next, I am going to use lda() to illustrate some of the basic calculations. The way that I use lda() here is not the way that I normally will use it. The purpose is to demonstrate the simplest way to implement DA so that I can show how the calculations are performed.

> library(MASS)

> DA1 <- lda(formula = good ~ distance + change + elap30 +

PAT + type + field + wind, data = placekick, CV =

FALSE, prior = c(1,1)/2)

> DA1

Call:

lda(good ~ distance + change + elap30 + PAT + type + field + wind, data = placekick, CV = FALSE)

Prior probabilities of groups:

0 1

0.5 0.5

Group means:

week distance change elap30 PAT type

0 9.895706 40.48466 0.4171779 10.75849 0.07361963 0.6809816

1 9.290808 25.87401 0.2305864 12.48986 0.59508716 0.7345483

field wind

0 0.4539877 0.11042945

1 0.4817750 0.07131537

Coefficients of linear discriminants:

LD1

week -0.012638520

distance -0.106246599

change -0.276001408

elap30 0.003397577

PAT -0.139069418

type 0.191509070

field -0.133839632

wind -0.448270472

> names(DA1)

[1] "prior" "counts" "means" "scaling" "lev"

[6] "svd" "N" "call" "terms" "xlevels"

> class(DA1)

[1] "lda"

> methods(class = "lda")

[1] coef model.frame pairs plot

[5] predict print

see '?methods' for accessing help and source code

> pred.resub <- predict(object = DA1)

> names(pred.resub)

[1] "class" "posterior" "x"

> head(pred.resub$posterior)

0 1

1 0.11002031 0.8899797

2 0.07753703 0.9224630

3 0.08764730 0.9123527

4 0.21463824 0.7853618

5 0.08577185 0.9142281

6 0.15073865 0.8492614

> head(pred.resub$class)

[1] 1 1 1 1 1 1

Levels: 0 1

> head(pred.resub$x) #A rescaled version of b’x, Values > 0

indicate a 1 classification

LD1

1 1.3324733

2 1.5783494

3 1.4932039

4 0.8268088

5 1.5082994

6 1.1019224

> class.lin.discrim.rule <- ifelse(test = pred.resub$x > 0, yes = 1, no = 0)

> rule2 <- ifelse(test = pred.resub$posterior[,1] < 0.5, yes = 1, no = 0)

> table(class.lin.discrim.rule, rule2) #Same

classifications

rule2

class.lin.discrim.rule 0 1

0 397 0

1 0 1028

1. CV = FALSE corresponds to using resubstitution; CV = TRUE corresponds to cross-validation
2. prior = c(1,1)/2 will be explained later in the notes; for now, think of this as specifying that we would expect 50% of all placekicks to be successes and 50% of all placekicks to be failures, which is unrealistic.
3. The “coefficients of linear discriminants” do not give **b** of **b**′**x** – k. Rather, lda() rescales the variables first and then essentially applies the methods discussed earlier to obtain coefficients. Pages 94-95 of Ripley’s (1996) book “Pattern Recognition and Neural Networks” provides the details for the calculations (Ripley is a co-author of lda()). Of course, the code within lda() shows how the calculations are performed too. The reasons given for this alternative calculation form is that it is computationally better. Overall, I am disappointed by what lda() gives though in the output because it is not the standard given by software packages or how DA is taught.
4. The predict() function provides the posterior probabilities, the resulting classifications, and the linear discriminant values for each observation.

Next, below are some of my own calculations using matrix algebra:

> obs1 <- t(as.matrix(placekick[1,-9])) #Force it to be an

actual column vector

> obs1

1

week 1.0000

distance 21.0000

change 1.0000

elap30 24.7167

PAT 0.0000

type 1.0000

field 1.0000

wind 0.0000

> #Matrix calculations

> pop0 <- placekick[placekick$good == 0,-9]

> pop1 <- placekick[placekick$good == 1,-9]

> N0 <- nrow(pop0)

> N1 <- nrow(pop1)

> sigma.hat0 <- cov(pop0)

> sigma.hat1 <- cov(pop1)

> sigma.hat.p <- ((N0 - 1)\*sigma.hat0 + (N1 –

1)\*sigma.hat1)/(N0 + N1 - 2)

> mu.hat0 <- as.matrix(colMeans(pop0)) #Force it to be an

actual column vector

> mu.hat1 <- as.matrix(colMeans(pop1))

> b <- solve(sigma.hat.p) %\*% (mu.hat0 - mu.hat1)

> k <- 0.5\*t(mu.hat0 - mu.hat1) %\*% solve(sigma.hat.p) %\*%

(mu.hat0 + mu.hat1)

> b

[,1]

week 0.019828728

distance 0.166691584

change 0.433021970

elap30 -0.005330499

PAT 0.218187703

type -0.300460912

field 0.209982629

wind 0.703297003

> k

[,1]

[1,] 5.821677

> t(b)%\*%obs1

1

[1,] 3.731143

> t(b)%\*%obs1 - k

1

[1,] -2.090534

> #Mahalanobis distances

> D0 <- t(obs1 - mu.hat0) %\*% solve(sigma.hat.p) %\*% (obs1 – mu.hat0)

> D1 <- t(obs1 - mu.hat1) %\*% solve(sigma.hat.p) %\*% (obs1 – mu.hat1)

> data.frame(D0, D1)

X1 X1.1

1 17.5536 13.37253

> #Posterior probabilities

> prob0 <- exp(-0.5\*D0)/(exp(-0.5\*D0)+exp(-0.5\*D1))

> prob1 <- exp(-0.5\*D1)/(exp(-0.5\*D0)+exp(-0.5\*D1))

> data.frame(prob0, prob1) #Matches R

X1 X1.1

1 0.1100203 0.8899797

Cross-validation can be used by specifying CV = TRUE in the lda() function. This ends up changing the structure of the items returned.

> DA2 <- lda(formula = good ~ week + distance + change +

elap30 + PAT + type + field + wind, data = placekick,

CV = TRUE, prior = c(1,1)/2)

> #DA2 #Don't print this! There is no nice summary printed

> names(DA2)

[1] "class" "posterior" "terms" "call"

[5] "xlevels"

> class(DA2) #No class is used! This is the default saying

a list is returned

[1] "list"

> head(DA2$posterior)

0 1

1 0.11038820 0.8896118

2 0.07755802 0.9224420

3 0.08775029 0.9122497

4 0.21572455 0.7842755

5 0.08586327 0.9141367

6 0.15151182 0.8484882

> head(DA2$class)

[1] 1 1 1 1 1 1

Levels: 0 1

All of the posterior probabilities are now in the object returned by lda(). The predict() function cannot be used in the situation to obtain them. Also, notice that the posterior probabilities are a “little” different than what we had before with resubstitution. Why do you think they are only a “little” different?

How good is the discriminant rule doing overall? We can examine its accuracy through my summarize.class() function:

> summarize.class <- function(original, classify) {

class.table <- table(original, classify)

numb <- rowSums(class.table)

prop <- round(class.table/numb,4)

overall <- round(sum(diag(class.table)) /

sum(class.table),4)

list(class.table = class.table, prop = prop,

overall.correct = overall)

}

> summarize.class(original = placekick$good, classify =

DA2$class)

$class.table

classify

original 0 1

0 124 39

1 274 988

$prop

classify

original 0 1

0 0.7607 0.2393

1 0.2171 0.7829

$overall.correct

[1] 0.7804

> summarize.class(original = placekick$good, classify =

pred.resub$class)

$class.table

classify

original 0 1

0 124 39

1 274 988

$prop

classify

original 0 1

0 0.7607 0.2393

1 0.2171 0.7829

$overall.correct

[1] 0.7804

Overall, we see that 124/(124 + 39) = 76% of failed placekicks are correctly predicted as being failures. Also, we see that 988/(274 + 988) = 78% of the successful placekicks are correctly predicted as being successes.

With respect to the validation data, we obtain the following results

> pred.valid <- predict(object = DA1, newdata = valid)

> head(pred.valid$posterior)

0 1

1 0.10194199 0.89805801

2 0.89708224 0.10291776

3 0.96220215 0.03779785

4 0.08565351 0.91434649

5 0.09353304 0.90646696

6 0.10282079 0.89717921

> summarize.class(original = valid$good, classify =

pred.valid$class)

$class.table

classify

original 0 1

0 30 8

1. 59 185

$prop

classify

original 0 1

0 0.7895 0.2105

1 0.2418 0.7582

$overall.correct

[1] 0.7624

The accuracy is very similar to the values given by resubstitution and cross-validation.

A more general discriminant rule

Sometimes one type of classification error is more important to avoid then another. Also, sometimes we expect the number of items in one population to be larger than the other population. This part of the notes discusses how we can handle these situations.

Example: HIV testing

Suppose a clinical trial is being conducted on a new HIV test. The test measures a number of different variables related to the presence of HIV. Using discriminant analysis, a rule is developed to classify the subjects as negative or positive. Suppose an older, more expensive test (“gold standard”) can be used to determine if someone is really HIV positive or not. Below are the possible outcomes:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | HIV test results | |
|  |  | Negative | Positive |
| HIV actual | No | Correct | Error |
| Yes | Error | Correct |

The misclassification of HIV actual = Yes as HIV test = Negative is probably much more serious than the other type of error in the table above. To reflect the differences in the seriousness of the errors, changes can be made to the discrimination rule.

Cost function (numerical penalty for a misclassification)

Let C(i|j) be the cost of classifying an observation in Πi when it should be in Πj. Note C(i|i) = 0.

Probability of classifying an observation

Let P(i|j) = probability of classifying an observation in Πi when it should be in Πj.

Prior Probabilities

Let pi be the prior probability that a randomly selected observation comes from Πi. This value represents a prior belief before the data is collected.

Average cost of misclassification

p1\*C(2|1)\*P(2|1) + p2\*C(1|2)\*P(1|2)

(prior prob. 1) \* (cost of 2|1 mis.) \* (prob. 2|1 happens) +

(prior prob. 2) \* (cost of 1|2 mis.) \* (prob. 1|2 happens)

This is known as the Bayes Risk, which is discussed in other statistics courses.

The prior distribution is denoted by pi, the loss function is C(j|i), and the probability density is P(j|i). Note that



where R(i) is the risk function.

Want the average cost of misclassification to be as small as possible!

A Bayes rule

A rule that minimizes the average cost of misclassification is called a Bayes rule.

For the situation here, choose Π1 if

p2\*f2(**x**;**θ**2)\*C(1|2) < p1\*f1(**x**;**θ**1)\*C(2|1)

and choose Π2 otherwise.

where f1(**x**; **θ**1) is the probability density function for Π1 and f2(**x**; **θ**2) is the probability density function for Π2 with parameter vectors **θ**1 and **θ**2, respectively.

Notes:

* + - This comes from the average cost of misclassification
    - If p1 = p2 and C(1|2) = C(2|1), then the Bayes rule is the likelihood rule (notice what falls out above).
    - If Π1 and Π2 both have multivariate normal populations and **Σ**1 = **Σ**2, then the Bayes rule is

Choose Π1 if  where



for i = 1,2, j = 1,2, and i ≠ j; otherwise choose Π2.

Notice the similarities to Mahalanobis distance.

R implementation

Prior probabilities can be specified in lda() with the prior argument. For example, we specified equal prior probabilities in the last example using

prior = c(1,1)/2

When specifying the prior probabilities, you need to specify them in the order recognized for the variable denoting the different populations. The lda() function treats this variable as being a *factor* type; i.e., a qualitative variable. R will order levels of a factor in alphabetical order with numbers appearing first (0, 1, 2, …, 9, …, a, A, b, B, …, z, Z). Below are some short examples (code in Factors.R):

> #Example #1

> set1 <- data.frame(cat.var = factor(c("D", "A", "A", "B", "D", "C", "1", "0", "1", "b")))

> set1

cat.var

1. D

2 A

3 A

4 B

5 D

6 C

7 1

8 0

9 1

10 b

> class(set1$cat.var)

[1] "factor"

> levels(set1$cat.var)

[1] "0" "1" "A" "b" "B" "C" "D"

> #Example #2 – shows what happens if you have a

numerical variable

> set2 <- data.frame(num.var = c(0,1,0,0,1))

> set2

num.var

1 0

2 1

3 0

4 0

5 1

> class(set2$num.var)

[1] "numeric"

> levels(set2$num.var)

NULL

> factor(set2$num.var)

[1] 0 1 0 0 1

Levels: 0 1

> levels(factor(set2$num.var))

[1] "0" "1"

For the placekicking example (placekick$good is 0 or 1), the prior probability for the failures (0) would need to be given first in a vector.

Generally, one will use prior probabilities equal to the proportions of observations from each population in the sample. This is the default in lda() so the argument does not need to be specified if this is what you want to use.

Specifying costs are not as easy. Costs can be included as “prior” probabilities using the following relationships:



One can then use  as the prior probabilities. Deciding what the costs should be is difficult as well and very dependent on the problem of interest. For this reason, we will not examine costs further in these notes.

Unequal Covariance Matrices

Suppose Π1 and Π2 both have multivariate normal populations and **Σ**1 ≠ **Σ**2. The Bayes rule becomes:

Choose Π1 if  where



otherwise choose Π2.

This is called a **quadratic discriminant analysis** because the corresponding discriminant rule written out involves quadratic elements of **x**. I did not include the rule here because 1) working with  is sufficient and 2) R does not provide it in the output.

We now have two types of DA:

* **Quadratic discriminant analysis** – Do not assume equality of the covariance matrices; use the R function qda()
* **Linear discriminant analysis** – Assume equality of the covariance matrices; use the R function lda()

How do you decide whether or not to use quadratic or linear discriminant analysis? Below is a discussion:

* Formally, one could perform a hypothesis test of H0:**Σ**1=**Σ**2 vs. Ha:**Σ**1≠**Σ**2. Unfortunately, this test relies on a multivariate normal distribution assumption. Also, it can conclude “reject H0” when the practical differences are not too much.
* One could say “use quadratic discriminant analysis” always to be “safe”. However, many more parameters need to be estimated with quadratic rather than linear discriminant analysis. Generally speaking in statistics, the more parameters you estimate, the more variability you may have in the results when applying the model to a new data set.
* Instead of solely relying on the results of the hypothesis test, Johnson’s textbook recommends performing discriminant analysis assuming **Σ**1=**Σ**2 and then assuming **Σ**1≠**Σ**2 to see which method finds the best rule.

Example: Placekicking data (PlacekickDA.R, Placekick.csv, valid.csv)

There are many more successful placekicks than failures (88.56% success). To incorporate this information into the DA, the prior probabilities are taken to be observed sampled values.

> mean(placekick$good)

[1] 0.885614

> DA3 <- lda(formula = good ~ week + distance + change +

elap30 + PAT + type + field + wind, data = placekick,

CV = FALSE)

> DA4 <- lda(formula = good ~ week + distance + change +

elap30 + PAT + type + field + wind, data = placekick,

CV = TRUE)

> head(DA4$posterior)

0 1

1 0.01577412 0.9842259

2 0.01074299 0.9892570

3 0.01227158 0.9877284

4 0.03430818 0.9656918

5 0.01198636 0.9880136

6 0.02254376 0.9774562

> #Cross-validation

> summarize.class(original = placekick$good, classify =

DA4$class)

$class.table

classify

original 0 1

0 60 103

1 77 1185

$prop

classify

original 0 1

0 0.3681 0.6319

1 0.0610 0.9390

$overall.correct

[1] 0.8737

> #Validation data

> pred.valid3 <- predict(object = DA3, newdata = valid)

> head(pred.valid3$posterior)

0 1

1 0.01444960 0.9855504

2 0.52959398 0.4704060

3 0.76678925 0.2332107

4 0.01195472 0.9880453

5 0.01315199 0.9868480

6 0.01458641 0.9854136

> summarize.class(original = valid$good, classify =

pred.valid3$class)

$class.table

classify

original 0 1

0 15 23

1 18 226

$prop

classify

original 0 1

0 0.3947 0.6053

1 0.0738 0.9262

$overall.correct

[1] 0.8546

Comments:

* Even though we generally only want the accuracy measures from cross-validation and from using the validation data, we still need to use lda() with CV = FALSE to use the predict() function with the validation data.
* There is a large increase in the proportion of correctly classified successes. There is a large decrease in the proportion of correctly classified failures. However, because the number of successes is so much larger than the number of failures, the overall correct classification rate has increased.

Next, I use quadratic discriminant analysis with priors that are equal to the proportion observed from each population. The qda() function works in a very similar manner as the lda() function.

> DA5 <- qda(formula = good ~ week + distance + change +

elap30 + PAT + type + field + wind, data = placekick,

CV = FALSE)

> DA5

Call:

qda(good ~ week + distance + change + elap30 + PAT + type + field + wind, data = placekick, CV = FALSE)

Prior probabilities of groups:

0 1

0.114386 0.885614

Group means:

week distance change elap30 PAT type

0 9.895706 40.48466 0.4171779 10.75849 0.07361963 0.6809816

1 9.290808 25.87401 0.2305864 12.48986 0.59508716 0.7345483

field wind

0 0.4539877 0.11042945

1 0.4817750 0.07131537

> DA6 <- qda(formula = good ~ week + distance + change +

elap30 + PAT + type + field + wind, data = placekick,

CV = TRUE)

> names(DA5)

[1] "prior" "counts" "means" "scaling" "ldet"

[6] "lev" "N" "call" "terms" "xlevels"

> names(DA6)

[1] "class" "posterior" "terms" "call"

[5] "xlevels"

> head(DA6$posterior)

0 1

1 0.2175043430 0.7824957

2 0.0684966384 0.9315034

3 0.0001330365 0.9998670

4 0.0553394298 0.9446606

5 0.0003076925 0.9996923

6 0.0420854539 0.9579145

> class(DA5)

[1] "qda"

> methods(class = "qda")

[1] model.frame predict print

see '?methods' for accessing help and source code

> class(DA6)

[1] "list"

> #Cross-validation

> summarize.class(original = placekick$good, classify =

DA6$class)

$class.table

classify

original 0 1

0 73 90

1 114 1148

$prop

classify

original 0 1

0 0.4479 0.5521

1 0.0903 0.9097

$overall.correct

[1] 0.8568

> #Validation data

> pred.valid5 <- predict(object = DA5, newdata = valid)

> head(pred.valid5$posterior)

0 1

1 0.0397702236 0.9602298

2 0.5485776273 0.4514224

3 0.8959444858 0.1040555

4 0.0003006699 0.9996993

5 0.0002069302 0.9997931

6 0.0001450762 0.9998549

> summarize.class(original = valid$good, classify =

pred.valid5$class)

$class.table

classify

original 0 1

0 17 21

1 24 220

$prop

classify

original 0 1

0 0.4474 0.5526

1 0.0984 0.9016

$overall.correct

[1] 0.8404

Summary of the classification error rates (always summarize results like this!)

| **DA** | **Cov. matrix** | **Priors** | **Accuracy method** | **Success** | **Failure** | **Overall** |
| --- | --- | --- | --- | --- | --- | --- |
| Linear | Equal | Equal | Cross-validation | 0.7607 | 0.7829 | 0.7804 |
|  |  |  | Validation | 0.7582 | 0.7895 | 0.7624 |
|  |  |  |  |  |  |  |
| Linear | Equal | Proportional | Cross-validation | 0.9390 | 0.3681 | 0.8737 |
|  |  |  | Validation | 0.9262 | 0.3947 | 0.8546 |
|  |  |  |  |  |  |  |
| Quadratic | Unequal | Proportional | Cross-validation | 0.9097 | 0.4479 | 0.8568 |
|  |  |  | Validation | 0.9016 | 0.4474 | 0.8404 |

Suppose all placekicks are classified as successes. Because there are 163 failures in the data set, the error rate is 163/1425 = 0.1144! This could be a place where a higher cost for misclassifying a failure as a success may be useful.

Below are PC score plots summarizing the classifications using linear DA with equal covariance matrices, proportional priors, and cross-validation (see code in program):

A graph with red and black dots

Description automatically generated

A diagram of a graph

Description automatically generated

Compare these plots to those obtained earlier. These previous plots showed that there tended to be more failures for observations with large PC#1. Generally, this is how the DA classified the observations. Due to the banding in these plots, we can also see that all PATs will be classified as successes. In the data set, there are missed PATs:

> xtabs(formula = ~ PAT + good, data = placekick)

good

PAT 0 1

0 151 511

1 12 751

so this shows us one place where the discriminant rule is in error. Note that xtabs() works very similar to table() but used a formula argument.

Question: Suppose a variable of interest is qualitative with 3 levels, say A, B, and C. How would you include this variable in the discriminant analysis?

The lda() and qda() functions automatically account for these variables.

For more on including qualitative variables in models, please see my regression analysis course available from [www.chrisbilder.com](http://www.chrisbilder.com).

**Variable selection**

An ultimate goal of most model-based methods is to find the “best” method that uses as few variables as possible. With respect to DA, we judge “best” by the classification accuracy. How does one select variables then in this setting? This is not a topic often discussed in multivariate analysis books. Johnson’s textbook does spend time on it, and he approaches the problem using classical variable selection procedures (forward, backward, and stepwise) through the use of ANCOVA models.

ANCOVA REVIEW

Consider the following one-way ANOVA model:

Yij = μ + αi + εij

where εij ~ ind. N(0,σ2), αi is the effect of treatment i, μ is the grand mean, and Yij is the response of the jth object to treatment i

Example: Placekicking

Let Yij be the distance of the jth placekick from the ith “population” (success or failure). Thus, Y11 = distance of placekick #1 for the failures population.

Let α1 = failure effect and α2 = success effect. If α1 = α2, there are no mean differences among the distances of the placekicks. In this case, would the distance be a good discriminator between the successes and failures?

A one-way ANCOVA model is

Yij = μ + αi + βixij + εij

where βi is a slope coefficient for covariate xij.

Example: Placekicking

Suppose xij is the elap30 variable. If α1 = α2, there are no mean differences among the distances when xij is taken into account. In this case, would the distance be a good discriminator between the kernel types?

The use of ANCOVA models then can help us determine which variables help to differentiate between the populations. Variable selection procedures can help us decide which variables are important.

Example: Placekicking

“Forward” selection begins by finding the variable that is the best discriminator among all the variables. Below are the ANOVA models that are considered in a simplistic notation form:

weekij = μ + αi + εij

distanceij = μ + αi + εij

windij = μ + αi + εij

The variable that produces the largest F statistic value (smallest p-value) for the test “Ho:α1=α2 vs. Ha: not all equal” is chosen as the “best” discriminator.

Suppose distance was chosen in the previous step. The next best discriminator is then chosen by examining the following models:

weekij = μ + αi + βidistanceij + εij

hrwij = μ + αi + βidistanceij + εij

The variable that produces the largest F statistic value (smallest p-value) for the test of “Ho:α1=α2 vs. Ha: not all equal” is chosen as the next best discriminator.

This process continues until all F-tests are “non-significant”.

Problems

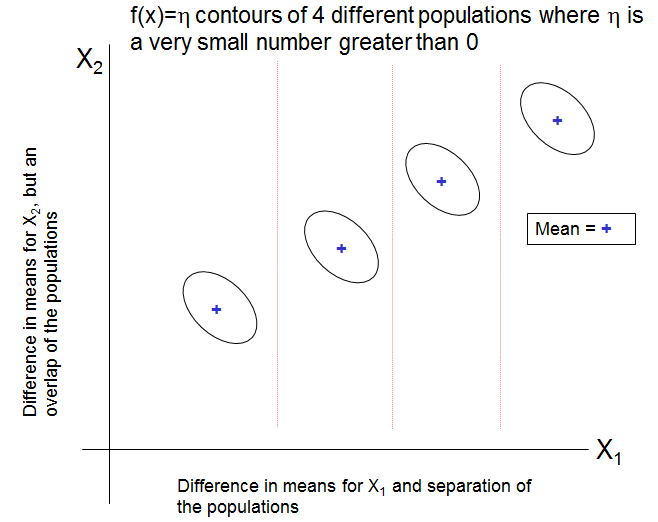
These procedures do not measure how well the variables actually discriminate! They are examining differences between means, not differences between entire populations. Means can be different, but populations can “overlap”.

Other problems can exist as well such as violating the distributional assumptions for the response and the issues with using classical variable selection procedures.

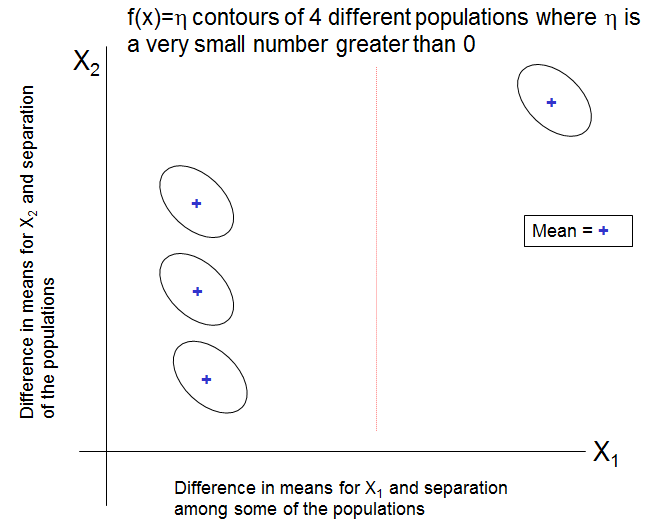
Example: Discrimination between 4 populations using 2 variables (x1 and x2)

While we have not directly discussed more than two populations, it is easier to present the ideas here with more than two. You will see shortly that the extension of DA to more than two populations is straightforward.

In the plot below, both x1 and x2 would probably be found to be “significant” by the variable selection procedures. Which variable will do the better job of discriminating between the 4 populations?



In the plot below, both x1 and x2 would probably be found to be “significant” by the variable selection procedures. Which variable do you think would be found as the “most significant” by the variable sections procedures?

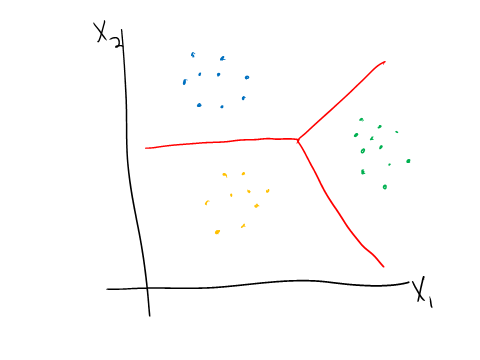


Comments

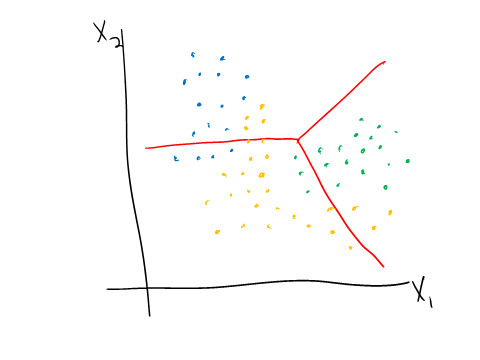
* Despite these problems, there can be some benefits to using the described variable selection procedure. If a reduced set of variables results in an accuracy level which is about the same or may be even better than when all variables are used, one has achieved at least some success with using the procedure ☺
* An overall issue also with using variable selection procedures for DA is whether or not using extra variables will really be detrimental to the classification process. “Extra” variables often will simply add some noise that will not decrease the classification accuracy by too much (or at all). The reason why variable selection is a much more important topic for courses regression analysis is that a main purpose of a regression model in those setting is to interpret the effect of a variable on the response. For our setting here, our top priority is finding a discriminant rule that produces as high accuracy as possible.
* Ideally, one could look at all possible combinations of the variables of interest and perform a DA with those variables. The “best” procedure is the one with largest accuracy. The problem with this approach is that there are 2P different discriminant analyses, where P is the number of variables of interest, that would need to be performed. Depending on the value of P and the sample size, it may not be practical to perform all of the analyses.

**Additional comments about DA**

* The discrimination rules given for two populations can be extended to having more than two populations. For example, if there were three populations, there would be three posterior probabilities to examine for each observation. The highest probability corresponds to the population that the observation is classified into.
* If there were two variables x1 and x2 used to differentiate between three populations (denoted by blue, green, and orange in the plot below), the discriminant rule would partition a plot of x1 vs. x2 as follows:



One can actually find equations for the red lines using the linear discriminant functions. The important point here is that there are THREE straight LINES used to differentiate between the different populations. In the above case, the DA works very well. However, suppose the following case occurs:



Note that the red lines may be not drawn in the exact same location as before, but I left them in the plot for demonstration purposes. Clearly, the observations form each population are separated. The problem is there are not a set of three straight lines that can be used to perfectly differentiate among the populations. For this particular instance then, more flexible classification methods could work much better.

* Suppose you have one data set that you would like to RANDOMLY split into model building and validation data sets. This can be done a couple of different ways. Below is a demonstration of the process using the placekick data (code is in PlacekickDA.R):

> #First, put the current model building and validation data into one data frame as would actually have occurred in

practice

> all.data <- rbind(placekick, valid)

> N.all <- nrow(all.data)

> #Obtain observation numbers

> set.seed(9819)

> obs.numb1 <- sample.int(n = N.all, size = round(N.all\*0.8))

> obs.numb2 <- (1:N.all)[-obs.numb]

> placekick1 <- placekick[obs.numb1,]

> placekick2 <- placekick[obs.numb2,]

> nrow(placekick1)

[1] 1366

> nrow(placekick2)

[1] 341

> head(placekick1)

week distance change elap30 PAT type field wind good

965 12 20 0 13.8500 1 1 1 0 1

1072 14 20 0 27.6667 1 1 1 0 1

119 2 54 0 10.5333 0 1 1 1 0

1315 16 20 0 14.3167 1 1 1 0 1

1537 8 20 1 15.7667 1 0 1 0 1

1634 14 28 0 3.0833 0 0 1 0 1

> head(placekick2)

week distance change elap30 PAT type field wind good

13 1 51 1 19.7833 0 1 1 0 0

14 1 20 0 29.1833 1 1 1 0 1

20 1 20 0 19.0333 1 1 1 0 1

27 1 20 0 22.1500 1 1 0 0 1

28 1 28 0 15.5833 0 1 0 0 1

30 1 20 0 4.0167 1 1 0 0 1

* Canonical discriminant analysis is somewhat similar to how I used PCA to help us see the DA results. In particular, new variables known as “canonical variables” are formed as linear combinations of the original variables. These variables are chosen in a way to maximize the differences between the populations. These new variables then are used in a DA. An advantage to this approach is that it helps one to see the results similar to what we already did.