Linear discriminant analysis

Evan Cummings
CSCI 548 – Douglas W. Raiford – Pattern Recognition

November 16, 2016

1 Iris data :

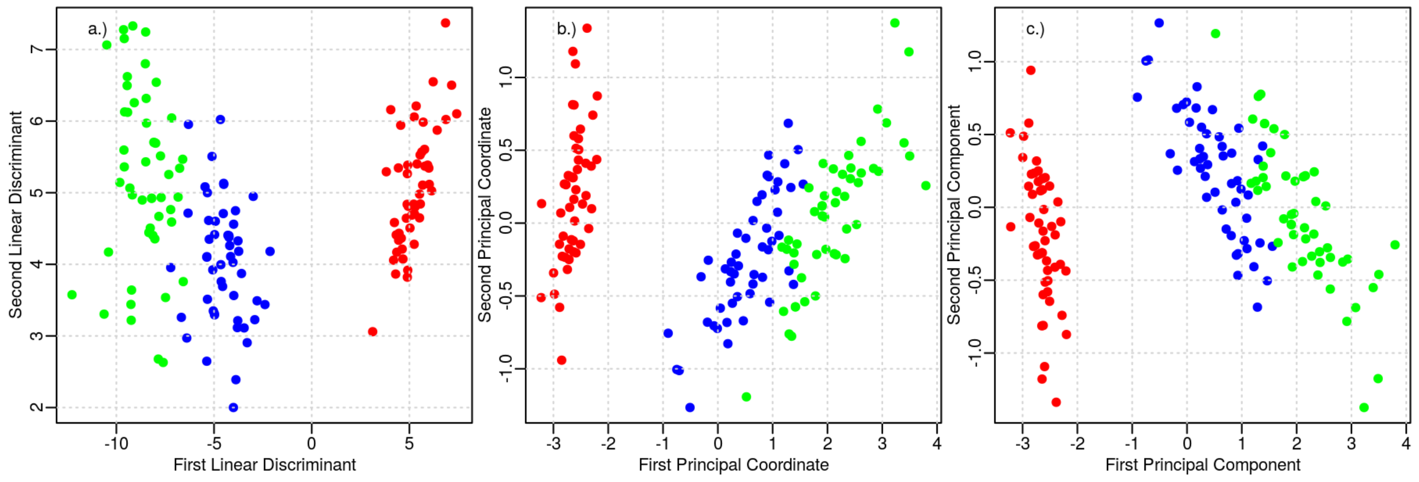

Figure 1: The Iris data with types Setosa (red), Versicolor (blue), and Virginica (green) projected onto the first two: a.) linear discriminants, b.) principal coordinates, and c.) principle components. Note the clear separation between Setosa from the other two types of Iris, for all three types of analyses. We used a random ≈ 83.3% of the `iris` dataset containing 125 elements to determine the first two linear discriminants, then used the resulting coefficients to predict the classes of the remaining 25 elements; the accuracy reported was ≈ 96%.

1.1 R source code :

```r
# iris data LDA project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

library(MASS)

# store the iris 'classes' :
c = iris[,5]

# store the iris 'data' :
d = iris[,seq(1,4)]
m = as.matrix(d)

# get training indicies :
t = sample(1:150, 125)

# perform LDA on d :
zlda = lda(d[t,], c[t])

# perform MDS on d :
zmds = cmdscale(dist(d))

# perform PCA on d :
zpca = prcomp(d)

# predict the test set -train :
p = predict(zlda, d[-t,])

# get the number predicted correctly :
correct = length(which(p$class == c[-t]))

# get the proportion correct :
prop = correct / length(c[-t])

# print the result to the screen :
cat("accuracy =", prop*100, "%\n")

# set the 1st linear discriminant :
xlda = m %*% zlda$scaling[,1]

# set the 2nd linear discriminant :
ylda = m %*% zlda$scaling[,2]

# store indexes of classes :
s  = which(c == 'setosa')
vc = which(c == 'versicolor')
vg = which(c == 'virginica')

# color vector :
cc     = as.vector(c)
cc[s]  = 'red'
cc[vc] = 'blue'
cc[vg] = 'green'

# plot them colored by class :
png('../doc/images/iris.png', res=200, width=9,
    height=3, units='in')
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(xlda, ylda, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "a.)", bty="n")
grid()
#legend('topright', levels(c),
#       col   = c('red', 'green', 'blue'),
#       pt.bg = c('red', 'green', 'blue'), pch = 21)

plot(zmds, col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Coordinate',
     ylab='Second Principal Coordinate', main='')
legend("topleft", "b.)", bty="n")
grid()

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
legend("topleft", "c.)", bty="n")
grid()

dev.off()
```

2 Fruit data :

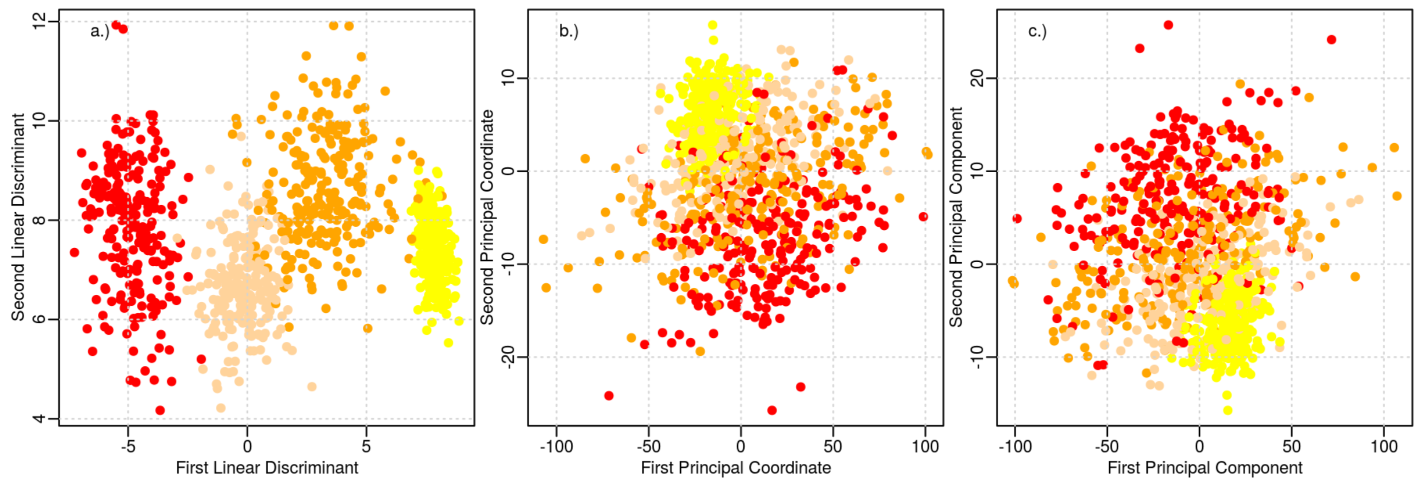

Figure 2: The Fruit data with types apples (red), lemons (yellow), oranges (orange), and peaches (burlywood1) projected onto the first two: a.) linear discriminants, b.) principal coordinates, and c.) principle components. Note the much clearer separation between classes using linear discriminant analysis from the other methods. We used a random ≈ 83.3% of the `fruit` dataset containing 833 elements to determine the first two linear discriminants, then used the resulting coefficients to predict the classes of the remaining 167 elements; the accuracy reported was ≈ 95.2%.

2.1 R source code :

```r
# fruit data LDA project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

library(MASS)

# read the variable back in :
f = read.csv("../../data/fruit.csv")

# store the fruit ’classes’ :
c = f[,5]

# store the fruit ’data’ :
d = f[,seq(1,4)]
m = as.matrix(d)

# get training indicies :
t = sample(1:1000, 833)

# perform LDA on d :
zlda = lda(d[t,], c[t])

# perform MDS on d :
zmds = cmdscale(dist(d))

# perform PCA on d :
zpca = prcomp(d)

# predict the test set -train :
p = predict(zlda, d[-t,])

# get the number predicted correctly :
correct = length(which(p$class == c[-t]))

# get the proportion correct :
prop = correct / length(c[-t])

# print the result to the screen :
cat("accuracy =", prop*100, "%\n")

# set the 1st linear discriminant :
xlda = m %*% zlda$scaling[,1]

# set the 2nd linear discriminant :
ylda = m %*% zlda$scaling[,2]

# store indexes of classes :
a = which(c == ’apple’)
l = which(c == ’lemon’)
o = which(c == ’orange’)
z = which(c == ’peach’)

# color vector :
cols  = c(’red’, ’yellow’, ’orange’, ’burlywood1’)
cc    = as.vector(c)
cc[a] = cols[1]
cc[l] = cols[2]
cc[o] = cols[3]
cc[z] = cols[4]

# plot them colored by class :
png(’../doc/images/fruit.png’, res=200, width=9,
    height=3, units=’in’)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(xlda, ylda, col=cc, bg=cc, type=’p’, pch=21,
     xlab=’First Linear Discriminant’,
     ylab=’Second Linear Discriminant’, main=’’)
legend("topleft", "a.)", bty="n")
grid()
#legend(’topright’, levels(c),
#        col   = c(’red’, ’green’, ’blue’),
#        pt.bg = c(’red’, ’green’, ’blue’), pch = 21)

plot(zmds, col=cc, bg=cc, type=’p’, pch=21,
     xlab=’First Principal Coordinate’,
     ylab=’Second Principal Coordinate’, main=’’)
legend("topleft", "b.)", bty="n")
grid()

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type=’p’, pch=21,
     xlab=’First Principal Component’,
     ylab=’Second Principal Component’, main=’’)
legend("topleft", "c.)", bty="n")
grid()

dev.off()
```

3 Mouse data :

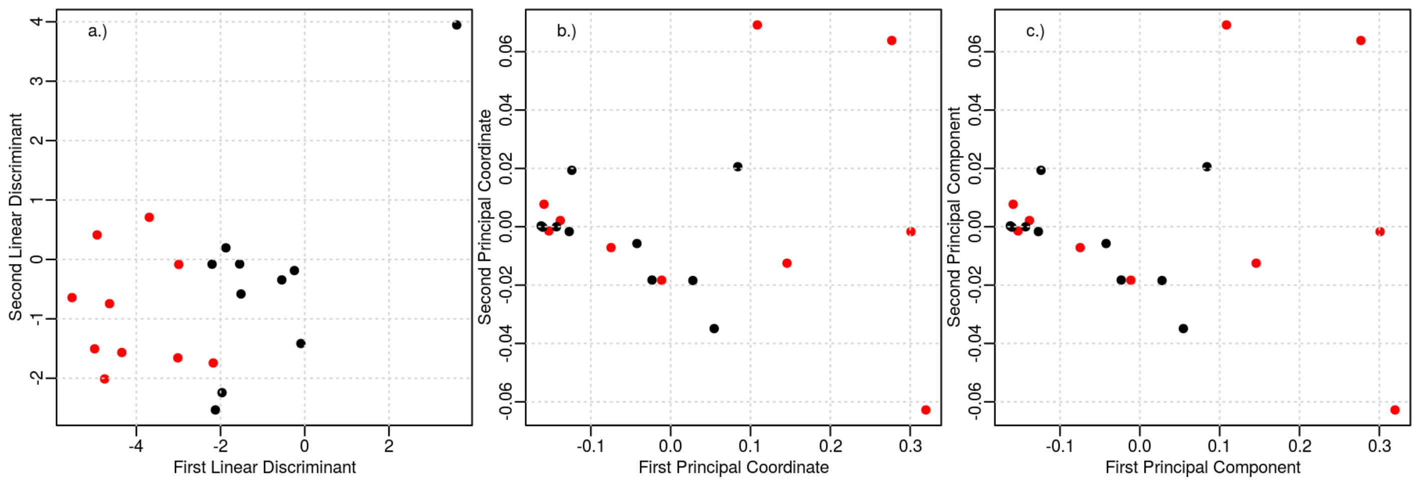

Figure 3: The Mouse data with experiment types proximal (black) and distal (red) projected onto the first two: a.) linear discriminants, b.) principal coordinates, and c.) principle components. Note the improved separation between experiments using linear discriminant analysis from the other methods. We used a random 18 elements of the `mouse` dataset containing 20 elements to determine the first two linear discriminants, then used the resulting coefficients to predict the classes of the remaining 2 elements; the accuracy reported was either 50% or 100%.

3.1 R source code :

```r
# mouse data LDA project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

library(MASS)

# read in the data :
f = read.csv("../../data/otu_table_L6.txt", sep="\t", row.names=1)

# remove and "Other" generas :
fn = f[-grep("Other", rownames(f)),]

# remove any rows with more than 6 zeros :
g = c()
for(i in 1:nrow(fn))
{
  # if too many zeros add to g :
  if (length(which(fn[i,] == 0)) >= 6)
  {
    g = c(g,i)
  }
}
fn = fn[-g,]

# get the experiment names :
e = colnames(fn)

# get the genera names :
g = rownames(fn)

# store indexes of proximal (P) and distal (D) experiments, and mouse types
# B and C :
P = grep("[A-Z]+P[0-9]", e)
D = grep("[A-Z]+D[0-9]", e)
B = grep("^B[A-Z]+[0-9]", e)
C = grep("^C[A-Z]+[0-9]", e)

# create the classes (proximal or distal) :
c    = 1:length(e)
c[P] = 'proximal'
c[D] = 'distal'

# transpose the data so each row is an experiment :
d = t(fn)
m = as.matrix(d)

# get training indicies :
t = sample(1:20, 18)
#t = 1:10

# perform LDA on d :
zlda = lda(d[t,], c[t], tol=0)

# perform MDS on d :
zmds = cmdscale(dist(d))

# perform PCA on d :
zpca = prcomp(d)

# predict the test set -train :
p = predict(zlda, d[-t,])

# get the number predicted correctly :
correct = length(which(p$class == c[-t]))

# get the proportion correct :
prop = correct / length(c[-t])

# print the result to the screen :
cat("accuracy =", prop*100, "%\n")

# set the 1st linear discriminant :
xlda = m %*% zlda$scaling[,1]

# set the 2nd linear discriminant :
ylda = m %*% zlda$scaling[,2]

#==============================================================================
# color vector :
cols  = c('black', 'red')
cc    = as.vector(e)
cc[P] = cols[1]
cc[D] = cols[2]

# plot them colored by experiment :
png('../doc/images/mouse_experiment.png', res=200, width=9,
    height=3, units='in')
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(xlda, ylda, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "a.)", bty="n")
grid()
#legend('topright', levels(c),
#       col   = c('red', 'green', 'blue'),
#       pt.bg = c('red', 'green', 'blue'), pch = 21)

plot(zmds, col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Coordinate',
     ylab='Second Principal Coordinate', main='')
legend("topleft", "b.)", bty="n")
grid()

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
legend("topleft", "c.)", bty="n")
grid()
dev.off()

#==============================================================================
# color vector :
cols  = c('purple', 'green')
cc    = as.vector(e)
cc[B] = cols[1]
cc[C] = cols[2]

# plot them colored by class :
png('../doc/images/mouse_class.png', res=200, width=9,
    height=3, units='in')
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(xlda, ylda, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "a.)", bty="n")
grid()
#legend('topright', levels(c),
#       col   = c('red', 'green', 'blue'),
#       pt.bg = c('red', 'green', 'blue'), pch = 21)

plot(zmds, col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Coordinate',
     ylab='Second Principal Coordinate', main='')
legend("topleft", "b.)", bty="n")
grid()

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
legend("topleft", "c.)", bty="n")
grid()
dev.off()
```

4 Tumor data :

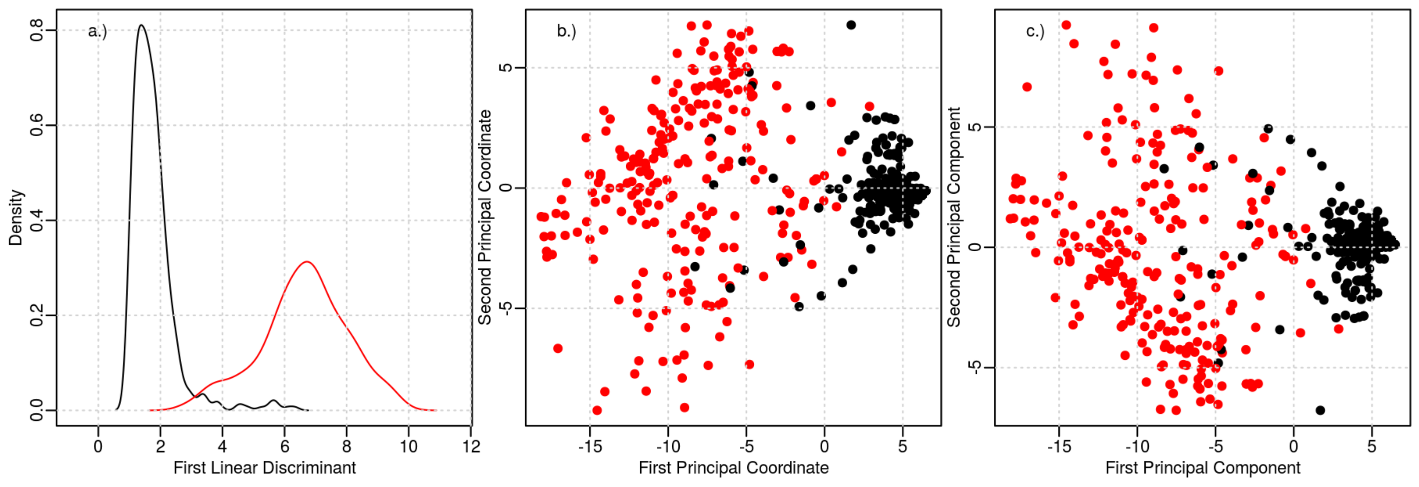

Figure 4: The Tumor data with types malignant (red) and benign (black) projected onto the first : a.) linear discriminant, then calculated density; b.) two principal coordinates, and c.) two principle components. Note that each analysis produces clear separation between classes, while the malignant cases' variance appears to be reduced. We used a random ≈ 85.8% of the `tumor` dataset containing 699 elements to determine the first linear discriminant, then used the resulting coefficients to predict the classes of the remaining 99 elements; the accuracy reported was ≈ 95%.

4.1 R source code :

```r
# tumor data LDA project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

library(MASS)

# read in the data :
f = read.csv("../../data/breast-cancer-wisconsin.data",
             header=FALSE, na.strings='?')

# get the sample names :
e = f[,1]

# get the classes :
c = f[,11]

# get just the data :
d     = f[,seq(2,10)]

# replace the NANs with the column mean :
for(i in 1:ncol(d))
{
  n_na         = which(!is.na(d[,i]))
  d[-n_na,i] = mean(d[n_na,i])
}
M = as.matrix(d)

# get training indicies :
t = sample(1:699, 600)

# perform LDA on d :
zlda = lda(d[t,], c[t])

# perform MDS on d :
zmds = cmdscale(dist(d))

# perform PCA on d :
zpca = prcomp(d)

# predict the test set -train :
p = predict(zlda, d[-t,])

# get the number predicted correctly :
correct = length(which(p$class == c[-t]))

# get the proportion correct :
prop = correct / length(c[-t])

# print the result to the screen :
cat("accuracy =", prop*100, "%\n")
```

```r
# set the 1st linear discriminant :
xlda = M %*% zlda$scaling[,1]

# create density to determine range of data :
dens   = density(xlda)
xrange = range(dens$x)

# color sample black for benign (b) and red for malignant (m) :
cols  = c('black', 'red')
cc    = as.vector(e)
b     = which(c == 2)
m     = which(c == 4)
cc[b] = cols[1]
cc[m] = cols[2]

# project just the benigns :
xb = M[b,] %*% zlda$scaling[,1]
db = density(xb)

# project just the malignant :
xm = M[m,] %*% zlda$scaling[,1]
dm = density(xm)

# plot them colored by experiment :
png('../doc/images/tumor.png', res=200, width=9,
    height=3, units='in')
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(db$x, db$y, col=cols[1], type='l', pch=21, xlim=xrange,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm$x, dm$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topleft", "a.)", bty="n")
grid()

plot(zmds, col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Coordinate',
     ylab='Second Principal Coordinate', main='')
legend("topleft", "b.)", bty="n")
grid()

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
legend("topleft", "c.)", bty="n")
grid()

dev.off()
```

5 Conclusions :

By including class information explicitly, LDA appears to be better able to differentiate between them; indeed, the LDA process results in a set of axes with *greatest separation* between classes. Furthermore, while the LDA classifier for the `mouse` data sometimes performs poorly depending on which elements were used to train the classifier, more often than not clear separation between classes is achieved (see Fig. 3). Therefore, with more data, it may be possible to generate a very successful LDA classifier for these data.