Statistics 452: Statistical Learning and Prediction

Chapter 4, Part 2: Linear Discriminant Analysis Classifier

Brad McNeney

2017-09-28

Linear Discriminant Analysis (LDA) Classifier

Logistic Regression vs LDA Classifieer

- Logistic regression models $Pr(Y = 1|X = x) = p(x)$.
  - More generally, polytomous regression models $Pr(Y = k|X = x)$.
- LDA models $Pr(X = x|Y = k)$, which relates to $Pr(Y = k|X = x)$ *via* Bayes' rule:

$$Pr(Y = k|X = x) = \frac{Pr(X = x|Y = k)Pr(Y = k)}{\sum_j Pr(X = x|Y = j)Pr(Y = j)}.$$

Notation and Terminology

- Notation: $f_k(X) = Pr(X = x|Y = k)$, $\pi_k = Pr(Y = k)$, $p_k(x) = Pr(X = x|Y = k)$, so that

$$p_k(x) = \frac{f_k(x)\pi_k}{\sum_j f_j(x)\pi_j}.$$

- Terminology:
  - π_k is the *prior* probability of class k
  - $p_k(x)$ is the *posterior* probability of class k given data x.

Bayes Classifier

- We saw in Chapter 2 that the classifier with the lowest error rate is the one that chooses the class with the highest posterior probability.
- If we have good estimates, $\hat{p}_k(x)$, we can choose the class with the highest **estimated** posterior probability.
- Estimate $p_k(x)$ by estimating
  - π_k: If training sample is a population random sample, use the sample proportions.
  - $f_k(x)$: LDA model is parametric, so we estimate f_k by estimating its parameters.

LDA Classifier: Model for $f_k(x)$ when $p = 1$

- Assume $f_k(x)$ is normal with mean μ_k and variance σ_k^2.
  - Then estimating f_k amounts to estimating μ_k and σ_k^2.
  - Further assume σ_k is the same for all k and let σ denote this common value.
- For a particular x, the class with highest posterior probability is the one with highest value of

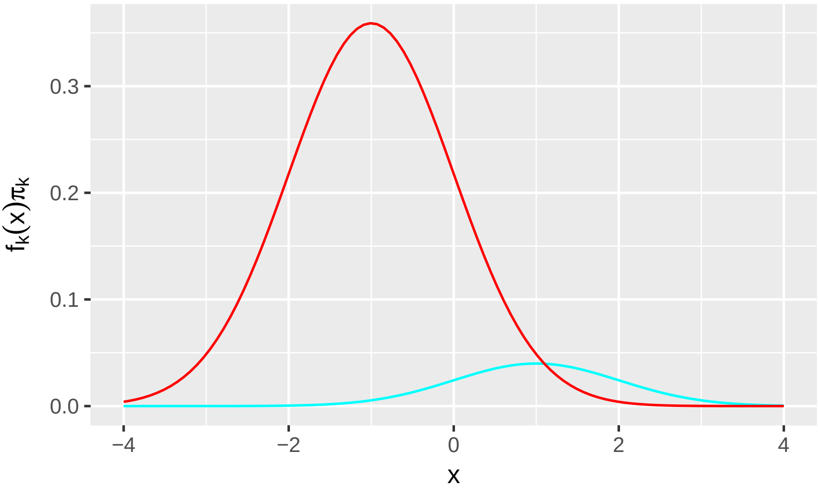
$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k.$$

Example

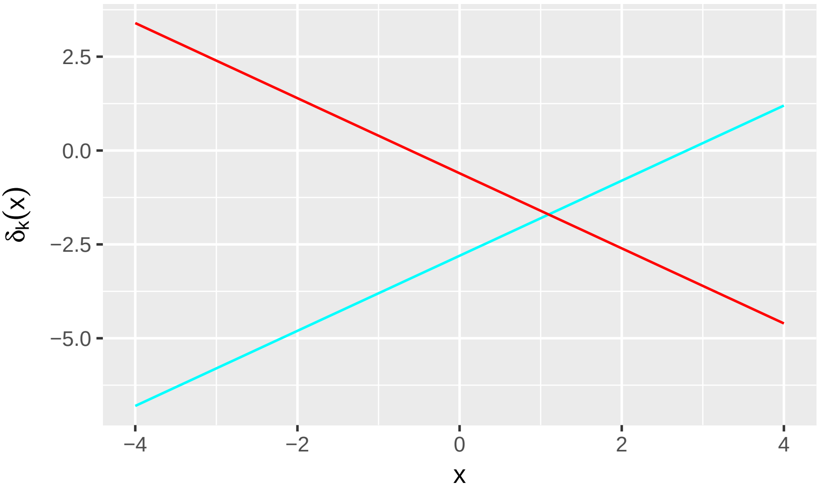
- Suppose two classes, $\pi_1 = 0.1$, $\pi_2 = 0.9$, $\mu_1 = 1$, $\mu_2 = -1$, $\sigma = 1$.

```
mu1<-1; mu2<-(-1); sigma<-1; pi1<-0.1;pi2<-0.9
x <-seq(from=-4,to=4,length=100)
f1 <- dnorm(x,mean=mu1,sd=sigma)
f2 <- dnorm(x,mean=mu2,sd=sigma)
delta <- function(x,mu,sigma,pi) {
  x*mu/sigma^2 - mu^2/(2*sigma^2) + log(pi)
}
delta1 <- delta(x,mu1,sigma,pi1)
delta2 <- delta(x,mu2,sigma,pi2)
dd <- data.frame(x=x,f1=f1,f2=f2,delta1=delta1,delta2=delta2)
```

```r
library(tidyverse)
ggplot(dd,aes(x=x))+
  geom_line(aes(y=f1*pi1),color="cyan")+
  geom_line(aes(y=f2*pi2),color="red") +
  labs(y=expression(f[k](x)*pi[k]))
```



```r
ggplot(dd,aes(x=x))+
  geom_line(aes(y=delta1),color="cyan")+
  geom_line(aes(y=delta2),color="red") +
  labs(y=expression(delta[k](x)))
```



Decision Boundary

- The point where the δ_k lines cross is the decision boundary.
- Can show that the boundary is

$$\frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2 \log(\pi_2/\pi_1)}{\mu_1 - \mu_2}.$$

- In this example the boundary is $(1 + (-1))/2 + \log(.9/.1)/2 \approx 1.1$

```
(mu1 + mu2)/2 + sigma^2*log(pi2/pi1)/(mu1-mu2)
```

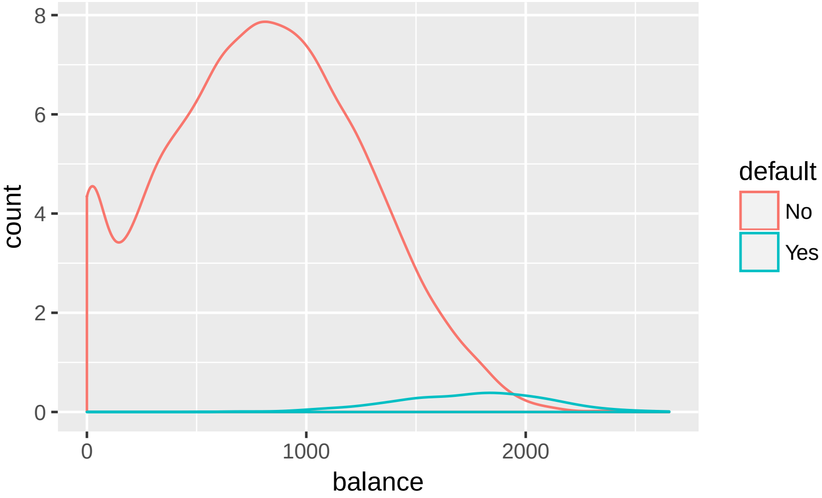
```
## [1] 1.098612
```

Estimated Distributions

- Replace means and variance in $\delta_k(x)$'s with estimates (equation 4.15 of text) to get the *discriminant functions* $\hat{\delta}_k$.
- Illustrate with `Default` data.

```
library(ISLR)
data(Default)
```

```r
ggplot(Default,aes(x=balance,color=default)) +
  geom_density(aes(y=..count..))
```



Software Note

- Specifying `color=default` groups the data by `default`.
- The density geom plots the estimated densities $\hat{f}_k(x)$ by default, but also computes the variable `..count..`, which for group k is $\hat{f}_k(x)n_k$.
- We are interested in $\hat{f}_k(x)\hat{\pi} = \hat{f}_k(x)n_k/n$, so `count` is proportional to what we want to plot.

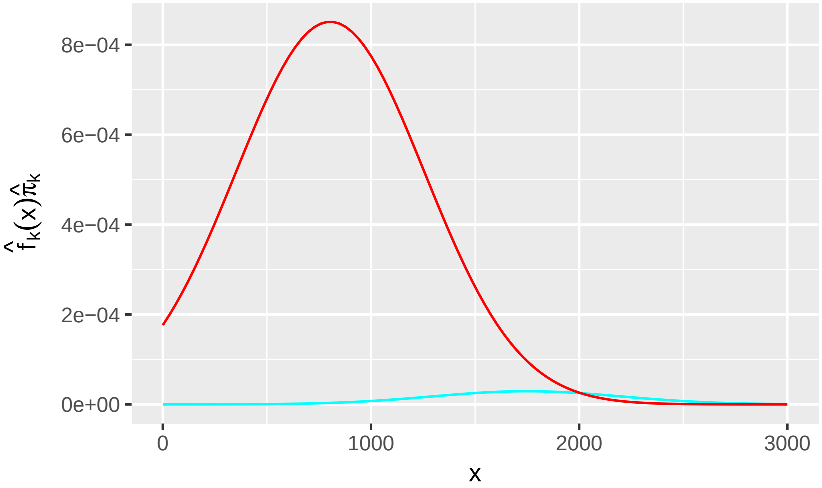
Estimated Densities and Discriminant Functions

```
n <- 10000; K <- 2
pi1 <- with(Default,mean(default=="Yes")); pi2 <- 1-pi1
defBalance <- with(Default,balance[default=="Yes"])
nodefBalance <- with(Default,balance[default=="No"])
mu1 <- mean(defBalance)
mu2 <- mean(nodefBalance)
sigma <- sqrt((sum((defBalance-mu1)^2) + sum((nodefBalance-mu2)^2))/(n-K))
#x <- seq(from=min(Default$balance),to=max(Default$balance),length=100)
with(Default,range(balance))
```

```
## [1]    0.000 2654.323
```

```
x <- seq(from=0,to=3000,length=100)
f1 <- dnorm(x,mean=mu1,sd=sigma)
f2 <- dnorm(x,mean=mu2,sd=sigma)
delta1 <- delta(x,mu1,sigma,pi1)
delta2 <- delta(x,mu2,sigma,pi2)
dd <- data.frame(x=x,f1=f1,f2=f2,delta1=delta1,delta2=delta2)
```

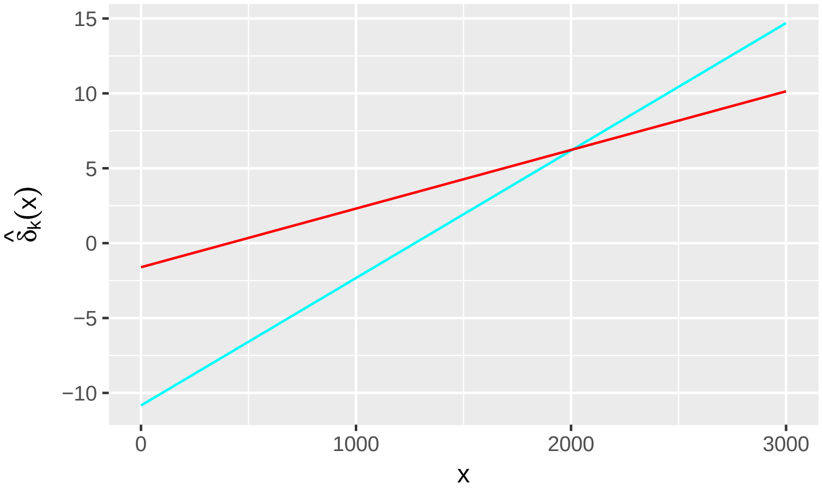
```r
ggplot(dd,aes(x=x))+ geom_line(aes(y=f1*pi1),color="cyan")+
  geom_line(aes(y=f2*pi2),color="red") +
  labs(y=expression(hat(f)[k](x)*hat(pi)[k]))
```




```r
min(x[f1*pi1>f2*pi2])
```

```
## [1] 2030.303
```

```r
ggplot(dd,aes(x=x))+
  geom_line(aes(y=delta1),color="cyan")+
  geom_line(aes(y=delta2),color="red") +
  labs(y=expression(hat(delta)[k](x)))
```



Decision Boundary for Default Data

```
(mu1 + mu2)/2 + sigma^2*log(pi2/pi1)/(mu1-mu2)
```

```
## [1] 2008.584
```

LDA Classifier in R

```
library(MASS)
ll <- lda(default ~ balance, data=Default)
preds <- predict(ll)
head(preds$posterior)
```

```
##          No         Yes
## 1 0.9972130 0.002786981
## 2 0.9958358 0.004164240
## 3 0.9865931 0.013406929
## 4 0.9988882 0.001111757
## 5 0.9963955 0.003604464
## 6 0.9933487 0.006651334
```

```
head(preds$class)
```

```
## [1] No No No No No No
## Levels: No Yes
```

LDA Classifier for $p > 1$

- Same basic idea of classifying based on highest posterior probabilities.
- Generalize the model for $f_k(X)$ from a Gaussian distibution to a *multivariate* Gaussian distribution.
- The mean in class k is now a vector μ_k and the variance is a matrix Σ.
- Details on the multivariate density and the resulting $\delta_k(x)$ are given on page 143 of the text.

LDA Classifier Using all Default Data

```
ll <- lda(default ~  student + balance + income, data=Default)
preds <- predict(ll)
head(preds$posterior)
```

```
##          No         Yes
## 1 0.9967765 0.003223517
## 2 0.9973105 0.002689531
## 3 0.9852914 0.014708600
## 4 0.9988157 0.001184329
## 5 0.9959768 0.004023242
## 6 0.9957918 0.004208244
```

Predictions *vs* True Default Status

- Tabulating predictions and true default status gives the following "confusion matrix":

```
Default <- data.frame(Default,predicted = preds$class)
xtabs( ~ predicted + default, data=Default)
```

```
##          default
## predicted   No  Yes
##       No  9645  254
##       Yes   22   79
```

- (Slight difference with results in text.)
- 9724 correctly predicted, 276 incorrectly predicted, so error rate is 2.76%.
- But 254/333, or 76% of defaulters missed.

Classification Error Rates

- Terminology is from diagnostic testing in medicine.
  - E.G., pap smear to screen for cervical cancer
- The test can be positive (T^+) or negative (T^-), and an individual may have the disease (D^+) or not (D^-):

		Disease Status	
		D^-	D^+
Test Result	T^-	True Negative (TN)	False Negative (FN)
	T^+	False Positive (FP)	True Positive (TP)
		TN+FP	FN+TP

- Replace test with prediction and true disease classification with true classification.

Sensitivity, Specificity, True- and False-Positive Rates

- The observed true positive rate (TPR) is the proportion of D^+ who are TP, or TP/(TP+FN).
  - TP/(TP+FN) is an estimate of $P(T^+ \mid D^+)$, known as the *sensitivity* of the test.
  - Sensitivity is only 79/333 or 23.7% for `Default` data.
- The observed true negative rate is the proportion of D^- who are TN, or TN/(TN+FP).
  - TN/(TN+FP) is an estimate of the true negative rate $P(T^- \mid D^-)$, known as the *specificity* of the test. Specificity is 9645/9667 or 99.8% for `Default` data.
  - The complement, FP/(TN+FP), is an estimate of the false positive rate (FPR) $P(T^+ \mid D^-)$. FPR is 22/9667 or 0.2% for `Default` data.

Receiver Operating Characteristic (ROC) Curves

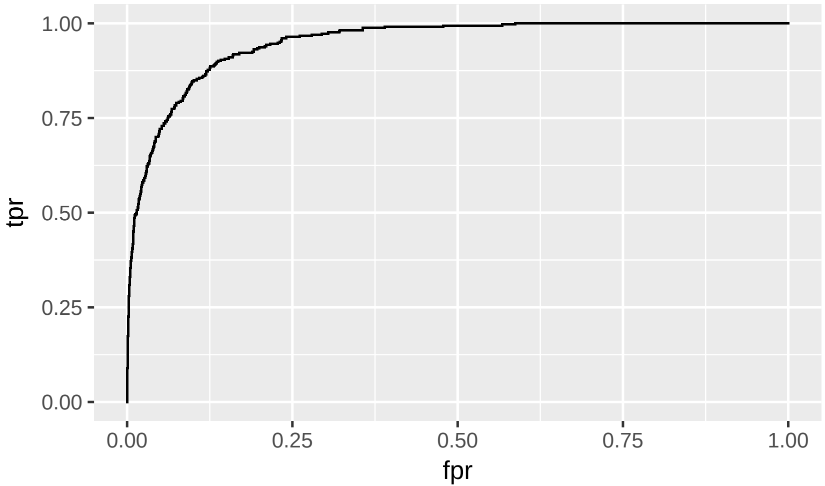
- The classification of a subject with $X = x_0$ as T^+ or T^- is made by comparing the posterior probability $Pr(T^+|X = x_0)$ to a threshold $t = 1/2$; e.g., $> t$ classify as T^+, $\leq t$ classify as T^-.
- Lowering the threshold will increase the number of T^+, and hence the TPR, $P(T^+ \mid D^+)$, and the FPR, $P(T^+ \mid D^-)$.
- The ROC curve is a plot plot of the TPR *versus* FPR as we vary t.
  - Try to get a sense of the trade-off between true- and false-positive rates, and possibly choose an "optimal" t.

ROC for LDA Classifier on Default Example

```r
library(AUC) # for the roc() function; install.packages("AUC") to install
library(broom) # for the tidy() function
posteriorYes <- preds$posterior[,"Yes"]
trueYes <- # require a binary factor with 1=Yes
  (Default$default=="Yes") %>% as.numeric() %>% factor()
ROCres <- roc(posteriorYes,trueYes)
tidyROCres <- tidy(ROCres)
```

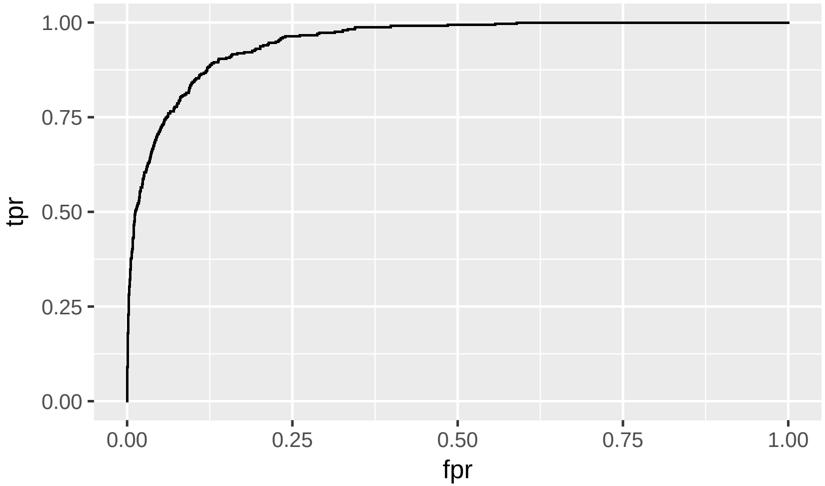
ROC Curve

```
ggplot(tidyROCres,aes(x=fpr,y=tpr)) + geom_point(pch=".")
```



ROC Curve for Logistic Regression

```
lfit <- glm(default ~ .,data=Default,family=binomial())
posteriorYesLogistic <- fitted(lfit)
ROClogist <- tidy(roc(posteriorYesLogistic,trueYes))
ggplot(ROClogist,aes(x=fpr,y=tpr)) + geom_point(pch=".")
```



ROC Interpretation

- The points on the plot represent the FPR/TPR combinations for each threshold.
- The ideal test threshold would yield a TPR of one and a FPR of zero, and so would appear in the upper-left corner of the ROC plot.
  - We usually select the threshold that is closest to the upper-left corner.
  - No obvious “best” threshold in this example. Perhaps the threshold that gives TPR of about 0.875 and FPR of about 0.125.

```
tidyROCres %>%
  filter(fpr >= 0.1249,fpr <= 0.1251)
```

```
##   instance     cutoff       fpr       tpr
## 1     7644 0.04572886 0.1249612 0.8768769
## 2     8518 0.04572196 0.1250647 0.8768769
## 3     8490 0.04569361 0.1250647 0.8798799
```

Re-classify

```r
n <- nrow(Default); thresh <- 0.0457
dclass <- rep("No",n); dclass[posteriorYes>thresh] <- "Yes"
Default <- data.frame(Default,prednew = dclass)
xtabs(~ prednew + default, data=Default)
```

```
##          default
## prednew    No  Yes
##      No  8458   41
##      Yes 1209  292
```

- Sensitivity is much better (about 87.7%).
- Many more false-positives (1209)

Area Under Curve (AUC)

- The area under the ROC curve is a measure of the overall performance of the classifier.
- If TPR jumps to near one with FPR remaining low, the performance is good.
  - This would give AUC near 1.
- A poor classifier would do no better than the null classifier, which for threshold t randomly assigns $t \times 100\%$ to be the success category and has AUC$=1/2$.

```
auc(ROCres)
```

```
## [1] 0.9495202
```

Quadratic Discriminant Analysis (QDA)

- Back to the $p = 1$ case.
- If we let the variances differ by class k, then it can be shown that the Bayes classifier assigns an observation with $X = x$ to the class with the largest

$$\delta_k(x) = -\frac{1}{2}\frac{(x - \mu_k)^2}{\sigma^2} - \frac{1}{2}\log\sigma_k^2 + \log\pi_k,$$

  which is a quadtratic function of x.
- See the text, page 149, for the multivariate version.

QDA for the Default Data

```r
qq <- qda(default ~ student + income + balance, data=Default)
preds <- predict(qq)
Default <- data.frame(Default,predicted = preds$class)
xtabs( ~ predicted + default, data=Default)
```

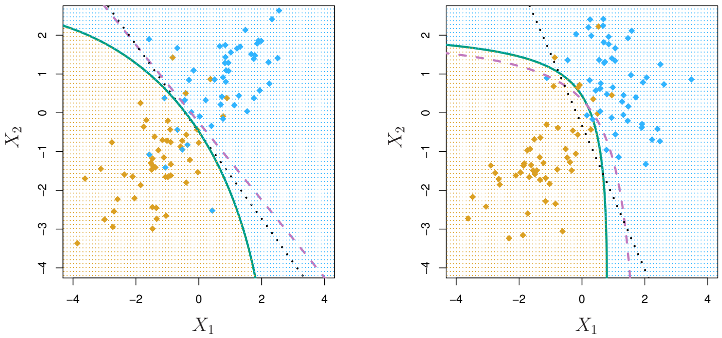
```
##          default
## predicted   No  Yes
##       No  9645  254
##       Yes   22   79
```

- Same as LDA for these data.

QDA *vs* LDA: Bias *vs* Variance

- When should we assume the K classes have a common variance matrix?
- LDA has fewer parameters and so lower variance.
  - When p is large relative to n, the number of parameters in QDA can become large, and variance of the posterior probabilities large.
- On the other hand, if the variances really **are** different, the LDA posterior probabilities can be biased.
- Recommendation from the text: If the training data set is large, use QDA. If not, use LDA.

QDA *vs* LDA: Decision Boundaries



- Text, Figure 4.9: $p=2), Bayes (purple dashed), LDA (black dotted), QDA (green solid).

Comparison of Classification Methods

- Performance depends on the nature of the true decision boundary (curve that separates the two classes).
- One can show that logistic regression leads to linear decision boundaries, just like LDA
  - Difference is in how the methods estimate the parameters of the boundaries.
  - Oftern give very similar results, as in the `Default` data.
- QDA gives quadratic boundaries.
- The KNN classifier is non-parametric and so there are no restrictions on the decision boundary.
- Performance depends on the nature of the true decision boundary.