Statistics 5401
27. More on Density-Based Classification

Gary W. Oehlert
School of Statistics
313B Ford Hall
612-625-1557
gary@stat.umn.edu

We are working on classification. We have two populations π_1 and π_2, each with prior probabilities p_1 and p_2. Within π_i, the covariates x follow the distribution $f_i(x)$. We have misclassification costs $c(1|2)$ and $c(2|1)$. The minimum expected cost classifier chooses π_1 when

$$\frac{f_1(x)}{f_2(x)} > \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

Suppose now that both populations are multivariate normal, but now we allow different means and variances in the two populations.
We still do minimum cost classification, but the (log) ratio of the densities is no longer a linear function of the covariates. It is now a quadratic function, and we get quadratic discrimination instead of linear discrimination.

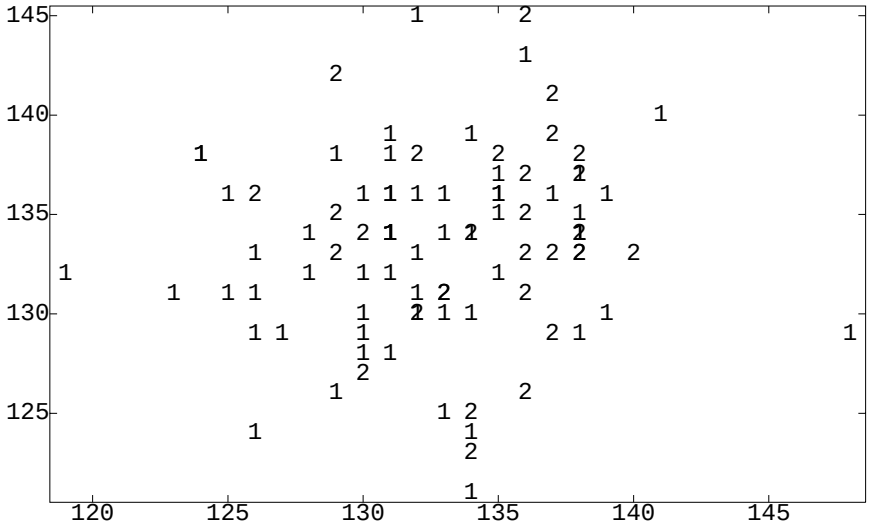
$$\begin{aligned}
&\log\left(\frac{f_1(x)}{f_2(x)}\right)\\
&= \frac{1}{2}x'(\Sigma_2^{-1}-\Sigma_1^{-1})x + (\mu_1'\Sigma_1^{-1}-\mu_2'\Sigma_2^{-1})x\\
&\quad -\frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) - \frac{1}{2}(\mu_1'\Sigma_1^{-1}\mu_1 - \mu_2'\Sigma_2^{-1}\mu_2)
\end{aligned}$$

Nothing really cancels
In practice, we must estimate the means and variances, so we classify into π_1 if

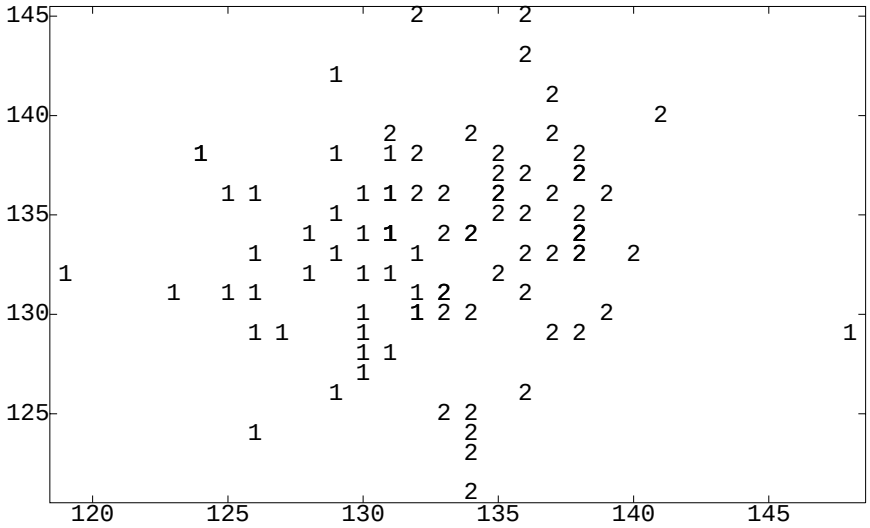
$$\begin{aligned}
&\frac{1}{2}x'(\mathbf{S}_2^{-1}-\mathbf{S}_1^{-1})x + (\overline{\mathbf{x}}_1'\mathbf{S}_1^{-1}-\overline{\mathbf{x}}_2'\mathbf{S}_2^{-1})x\\
&-\frac{1}{2}\ln\left(\frac{|\mathbf{S}_1|}{|\mathbf{S}_2|}\right) - \frac{1}{2}(\overline{\mathbf{x}}_1'\mathbf{S}_1^{-1}\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2'\mathbf{S}_2^{-1}\overline{\mathbf{x}}_2)\\
&\qquad\qquad \geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]
\end{aligned}$$

```
Cmd> X <- matrix(vecread(""),5)'
Read from file "~/JW5data/T6-13.DAT"

Cmd> three <- X[,5]==3

Cmd> gps <- three+1

Cmd> X <- X[,-5]
```

```
Cmd> X2 <- X[,run(2)]

Cmd> qout <- discrimquad(gps,X2)

Cmd> qout
component: Q
  gps 1
             (1)          (2)
(1)    -0.020576    0.0023015
(2)    0.0023015    -0.024484
  gps 2
             (1)          (2)
(1)    -0.041341    0.0013113
(2)    0.0013113    -0.020214
component: L
  gps 1
          (1)          (2)
    -0.034668   -0.0066206
  gps 2
          (1)          (2)
      0.14218     0.012973
component: addcon
           gps 1        gps 2
(1)      -3.1245      -2.9776
component: grandmean
          (1)          (2)
       132.73       133.37

Cmd> pp <- probsquad(X2,qout)

Cmd> pp
            gps 1        gps 2
(1)       0.50589      0.49411
(2)       0.92561      0.07439
(3)       0.56105      0.43895
(4)       0.99808    0.0019191
(5)       0.26389      0.73611
...

Cmd> pred <- pp[,2] > .5;pred <- pred+1

Cmd> chplot(X2[,1],X2[,2],gps)
```




```
Cmd> chplot(X2[,1],X2[,2],pred)
```




```
Cmd> probsquad(X2,qout,prior:vector(.9,.1))
            gps 1        gps 2
(1)        0.9021     0.097899
(2)       0.99115    0.0088508
(3)       0.92002     0.079978
(4)       0.99979    0.0002136
(5)        0.7634       0.2366
(6)       0.83274      0.16726
...
```

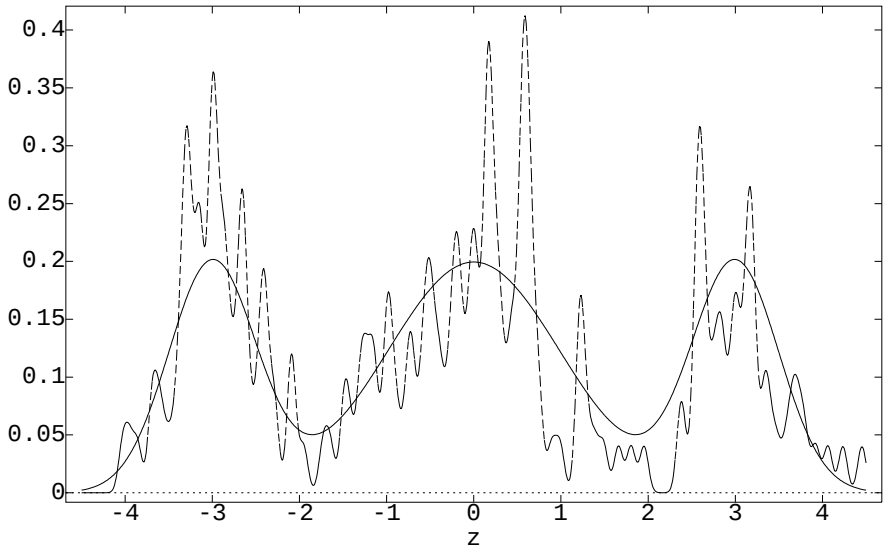
Nonparametric estimates.
With (very) large data sets, we can estimate $f_i()$ and not just assume normality.
Suppose that x is p-dimensional, and let $w()$ be a density. For example, $w()$ could be the standard multivariate normal. Let $x_{1i}, i = 1, \ldots, n$ be a sample from $f_1()$, ie, from π_1. Similarly, $x_{2i}, i = 1, \ldots, m$ is a sample from π_1.
Let λ be some scalar to be chosen later.
Estimate $f_1()$ via

$$\hat{f}_{1,\lambda}(x) = \frac{1}{n\lambda^p} \sum_{i=1}^{n} w\left(\frac{x - x_{1i}}{\lambda}\right)$$

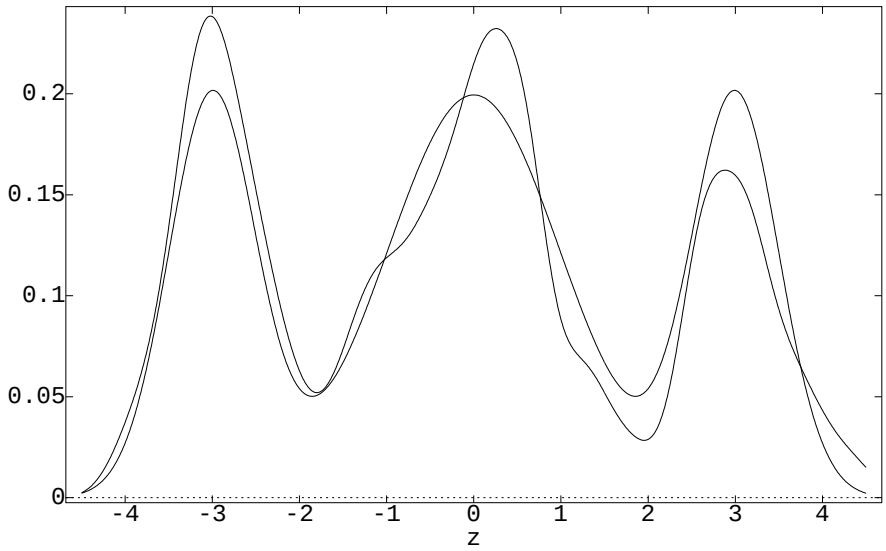
It is easy to see that this is a density (it's the average of n densities).
When λ is large, this estimate oversmooths; when λ is small, it undersmooths. Choosing the right value of λ is a difficult task.

```
Cmd> x <- rep(0,200)

Cmd> for(i,run(200)) {
if(runi(1) < .5) {
x[i] <- rnorm(1)
} else {
tmp <- .5*rnorm(1)+3
if(runi(1) < .5) {
tmp <- -tmp
}
x[i] <- tmp
}
;;
}

Cmd> z <- run(-4.5,4.5,.01)

Cmd> f <- .25*exp(-(z- (-3))^2/(2*.25))/sqrt(2*PI*.25) + .5*exp(-(z- (0))^2/(2
.25*exp(-(z- (3))^2/(2*.25))/sqrt(2*PI*.25)

Cmd> kden1 <- macro("
@x <- $1
@lam <- $2
@x0 <- $3
@tmp <- sum(exp(-(@x-@x0')^2/(2*@lam^2)))
vector(@tmp)/length(@x)/sqrt(2*PI)/@lam")

Cmd> fp05 <- kden1(x,.05,z)

Cmd> fp25 <- kden1(x,.25,z)

Cmd> fp5 <- kden1(x,.5,z)

Cmd> f1 <- kden1(x,1,z)
```

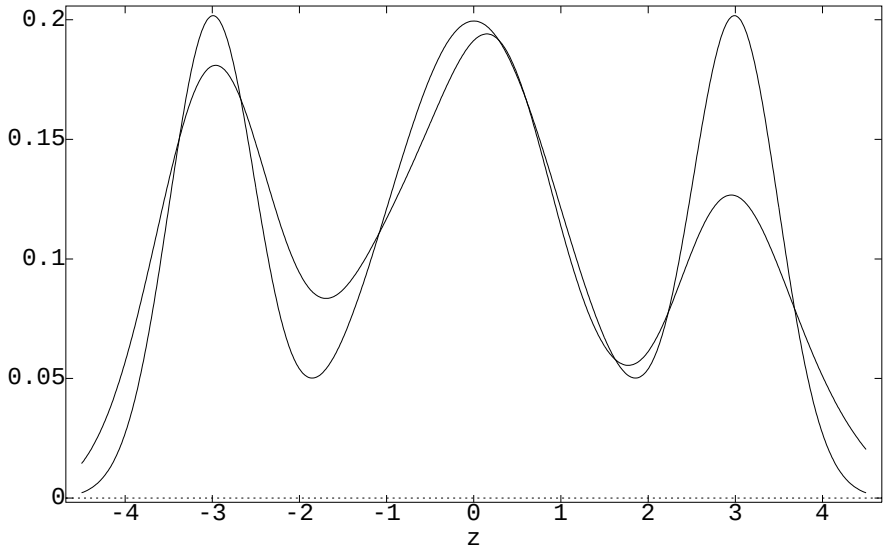
```
Cmd> lineplot(z,hconcat(f,fp05),yaxis:F)
```



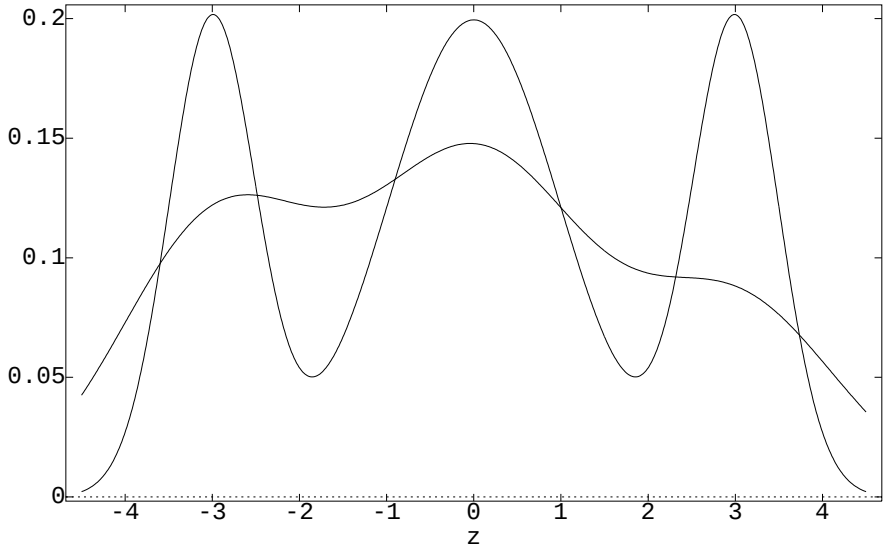

```
Cmd> lineplot(z,hconcat(f,fp25),yaxis:F)
```




```
Cmd> lineplot(z,hconcat(f,fp5),yaxis:F)
```




```
Cmd> lineplot(z,hconcat(f,f1),yaxis:F)
```



Evaluating a sample classifier.
One way is Total Probability of Misclassification (TPM). This is the probability that a randomly chosen object from the full population is misclassified.

$$TPM = p_1 P(2|1) + p_2 P(1|2)$$

where $P(2|1)$ is the probability that an observation from π_1 is classified into π_2. Similarly for $P(1|2)$.

$$TPM = p_1 \int_{R_2} f_1(x)dx + p_2 \int_{R_1} f_2(x)dx$$

The minimum value of TPM is the Optimum Error Rate (OER). We can never know this in practice.

Also, we never know R_1 correctly in practice, so we must use a sample-based classification rule. This gives us an Actual Error Rate (AER):

$$AER = p_1 \int_{\hat{R}_2} f_1(x)dx + p_2 \int_{\hat{R}_1} f_2(x)dx$$

We never know the AER either, because we don't know the $f_i()$s exactly.
Well, what *can* we compute? We can compute the Apparent Error Rate (APER), which is the error rate we get when we apply our sample-based classifier to our sample.
APER is easy to compute, but it is usually an optimistic (too low) estimate of AER.

		Predicted π_1	Predicted π_2	Total
Actual	π_1	n_{1c}	n_{1m}	n_1
	π_2	n_{2m}	n_{2c}	n_2

$$APER = \frac{n_{1m} + n_{2m}}{n_1 + n_2}$$

Some of these fractions have special names. Let π_2 be those patients with a disease; π_1 is those without.
The sensitivity is n_{2c}/n_2.
The specificity is n_{1c}/n_1.
Combine with prevalence and use Bayes's rule to get positive predictive accuracy (the probability that a person classified as with disease actually has the disease).
A better estimate of Actual Error Rate can be obtained by using Cross Validation (leave-one-out, also sometimes called Jackknifing).
For every subject, create a classification rule using all the data except that subject, and determine whether the subject is correctly classified by the leave-one-out procedure.
The APER for the leave-one-out is the cross-validated estimate of AER.

```
Cmd> tmp <- X2%*%discrim(gps,X2)$coefs\
+discrim(gps,X2)$addcon

Cmd> tmp <- tmp'

Cmd> tmp <- tmp-max(tmp)

Cmd> tmp <- exp(tmp)/sum(exp(tmp))

Cmd> tmp <- tmp';tmp
             gps1         gps2
(1)       0.54602      0.45398
(2)        0.7449       0.2551
(3)       0.57445      0.42555
(4)       0.85871      0.14129
....

Cmd> pred <- tmp[,2]>.5;pred <- pred+1
```

```
Cmd> out<-jackknife(gps,X2);out
 (1,1)      0.54067      0.45933            1
 (2,1)      0.74036      0.25964            1
 (3,1)      0.57331      0.42669            1
 (4,1)       0.8602       0.1398            1
...

Cmd> tabs(,gps,out[,3])
(1,1)           37           23
(2,1)           12           18

Cmd> tabs(,gps,pred)
(1,1)           39           21
(2,1)           12           18
```