Displays for Statistics 5401

Lecture 36

December 2, 2005

Christopher Bingham, Instructor
612-625-1024

Class Web Page
http://www.stat.umn.edu/~kb/classes/5401

Posterior probabilities

$g = 2$ case with $\Sigma_1 = \Sigma_2 = \Sigma$:

The posterior probability of π_i conditional on $\mathbf{x}$ is $P(\pi_i \mid \mathbf{x}) = p_i f_i(\mathbf{x}) / \sum_j p_j f_j(\mathbf{x})$

$= p_i e^{-(\mathbf{x} - \mu_i)' \Sigma^{-1} (\mathbf{x} - \mu_i)/2} / g(\mathbf{x})$, $i = 1, 2$

where

$$g(\mathbf{x}) = p_1 e^{-(\mathbf{x} - \mu_1)' \Sigma^{-1} (\mathbf{x} - \mu_1)/2} + p_2 e^{-(\mathbf{x} - \mu_2)' \Sigma^{-1} (\mathbf{x} - \mu_2)/2}$$

Factor out $e^{-\mathbf{x}' \Sigma^{-1} \mathbf{x}/2}$ and cancel and you get

$P(\pi_i \mid \mathbf{x}) = p_i e^{\ell_i' \mathbf{x} - c_i} / h(\mathbf{x})$, $i = 1, 2$.

- $h(\mathbf{x}) = p_1 e^{\ell_1' \mathbf{x} - c_1} + p_2 e^{\ell_2' \mathbf{x} - c_2}$
- $\ell_i = \Sigma^{-1} \mu_i$, $c_i = \ell_i' \mu_i / 2$

Divide top and bottom by $e^{\ell_2' \mathbf{x} - c_2}$:

$P(\pi_1 \mid \mathbf{x}) = p_1 e^{\ell' \mathbf{x} - c} / (p_1 e^{\ell' \mathbf{x} - c} + p_2)$

$P(\pi_2 \mid \mathbf{x}) = p_2 / (p_1 e^{\ell' \mathbf{x} - c} + p_2)$

with $\ell = \ell_1 - \ell_2 = \Sigma^{-1}(\mu_1 - \mu_2)$,

$c = c_1 - c_2 = \ell'(\mu_1 + \mu_2)/2$

These posterior probabilities depend on $\mathbf{x}$ through $\ell' \mathbf{x} - c$.

A computational problem

$\ell_i' \mathbf{x} - c_i$ can be so large that you can't compute $\exp(\ell_i' \mathbf{x} - c_i)$ because of overflow.

```
Cmd> exp(vector(709.782,709.783)) #709.782 ~ log(2^1024)
WARNING: exp(x) with result too large set to MISSING
(1) 1.7964e+308     MISSING
```

Solution:

Replace $\ell_i' \mathbf{x} - c_i$ by $\ell_i' \mathbf{x} - c_i - K(\mathbf{x})$, where $K(\mathbf{x})$ may depend on $\mathbf{x}$ but not on i.

Because $e^{-K(\mathbf{x})}$ cancels this leaves the ratio unchanged:

$$P(\pi_i \mid \mathbf{x}) = \frac{p_i e^{\ell_i' \mathbf{x} - c_i - K(\mathbf{x})}}{p_1 e^{\ell_1' \mathbf{x} - c_1 - K(\mathbf{x})} + p_2 e^{\ell_2' \mathbf{x} - c_2 - K(\mathbf{x})}}$$

Possible choices for $K(\mathbf{x})$:

- $K(\mathbf{x}) = \ell_1' \mathbf{x}$ or $K(\mathbf{x}) = \ell_2' \mathbf{x}$
- $K(\mathbf{x}) = \max\{\ell_1' \mathbf{x} - c_1, \ell_2' \mathbf{x} - c_2\}$ so that $\ell_i' \mathbf{x} - c_i - K(\mathbf{x}) \le 0$, $i = 1, 2$.

Why are posterior probabilities $P(\pi_j \mid \mathbf{x})$ *important*?

1. When costs differ, you need $P(\pi_i \mid \mathbf{x})$ to do minimum ECM classification.

2. They tell you how sure you should be that the classification being made is correct.

Examples with $g = 2$, with $C(1 \mid 2) = C(2 \mid 1)$ so that the minimum ECM rule is the minimum TPM rule. Suppose

- $P(\pi_1 \mid \mathbf{x}_1) = .51 \Rightarrow$ classify $\mathbf{x}_1$ as from π_1
- $P(\pi_1 \mid \mathbf{x}_2) = .97 \Rightarrow$ classify $\mathbf{x}_2$ as from π_1

because, in both cases,

$$P(\pi_1 \mid \mathbf{x}_i) > P(\pi_2 \mid \mathbf{x}_i) = 1 - P(\pi_1 \mid \mathbf{x}_i)$$

But $P(\pi_1 \mid \mathbf{x}_1) = .51$ means there a 49% posterior probability that π_1 is the wrong choice.

When $P(\pi_1 \mid \mathbf{x}_2) = .97$, there's only a 3% chance you are wrong.

Multi-group (g > 2) normal case

First look at the equal $\boldsymbol{\Sigma}$ case:

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \ldots = \boldsymbol{\Sigma}_g = \boldsymbol{\Sigma}$$

You can again do minimum TPM classification with linear functions $\boldsymbol{\ell}_i'\mathbf{x}$.

The posterior probabilities are

$$P(\pi_i \mid \mathbf{x}) = \frac{p_i e^{\boldsymbol{\ell}_i'\mathbf{x} - c_i - K(\mathbf{x})}}{\sum_{1 \le j \le g} p_j e^{\boldsymbol{\ell}_j'\mathbf{x} - c_j - K(\mathbf{x})}}$$

where

- $\boldsymbol{\ell}_i \equiv \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i$, i = 1,...,g
- $c_i \equiv \boldsymbol{\mu}_i'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i/2 = \boldsymbol{\ell}_i'\boldsymbol{\mu}_i/2$, i = 1,...,g
- $K(\mathbf{x})$ is an function of x chosen so that $\exp(\boldsymbol{\ell}_i'\mathbf{x} - c_i - K(\mathbf{x}))$ is computable for all i, for example,

  $$K(\mathbf{x}) = \max_{1 \le i \le g}\{\boldsymbol{\ell}_i'\mathbf{x} - c_i\}$$

  for which choice

  $$\boldsymbol{\ell}_i'\mathbf{x} - c_i - K(\mathbf{x}) \le 0$$

Rule for <u>minimum TPM classification</u>:

Select the population with largest $P(\pi_i \mid \mathbf{x})$.

By comparing the values of g linear functions, you can use $\hat{\pi}_{\text{min TPM}}$ without computing $P(\pi_i \mid \mathbf{x})$:

Select the population with the largest $\boldsymbol{\ell}_i'\mathbf{x} - c_i + \log(p_i)$ = linear function of $\mathbf{x}$

$P(\pi_i \mid \mathbf{x})$ measures the strength of the evidence in favor of π_i, i = 1,...,g.

For minimum ECM classification

- Compute posterior probabilities $P(\pi_j \mid \mathbf{x})$, j = 1,...,g
- For each π_i, compute the <u>posterior expected cost of misclassification</u>:

  $$ECM_i(\mathbf{x}) = \sum_{1 \le j \le g} P(\pi_j \mid \mathbf{x}) C(i \mid j)$$

- Select the population with <u>smallest</u> ECM_i.

When $p_1 = p_2 = \ldots = p_g = 1/g$, the minimum TPM rule for g MVN populations with $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \ldots = \boldsymbol{\Sigma}_g$ takes a particularly simple form:

- Select the population with the *smallest* value of

  $$(\mathbf{x} - \boldsymbol{\mu}_i)'\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i),$$

  the <u>Mahalanobis distance</u> between $\mathbf{x}$ and $\boldsymbol{\mu}_i$.

Thus, in this case, the minimum TPM rule is

- Select the population whose mean is "closest" to $\mathbf{x}$, using Mahalanobis distance.

Partitioning of observation space

Any classification rule $\hat{\pi}(\mathbf{x})$ implicitly partitions the p-dimensional space of possible observations into g regions

$$R_i = \{\mathbf{x} \mid \hat{\pi}(\mathbf{x}) = \pi_i\},\ i = 1, \ldots, g.$$

In general, region R_i can be <u>fragmented</u>, (disconnected pieces), can have <u>holes</u>, and can be <u>unbounded</u>.

When a rule is based on comparing functions of <u>linear functions</u> of $\mathbf{x}$, the boundary between any adjacent R_i and R_j is linear, that is, it is

- a line when p = 2
- a 2-dimensional plane when p = 3
- a p-1 dimensional plane when $p \ge 2$.

Each region R_i consists of one piece (is connected) and has no holes. R_i can be unbounded.

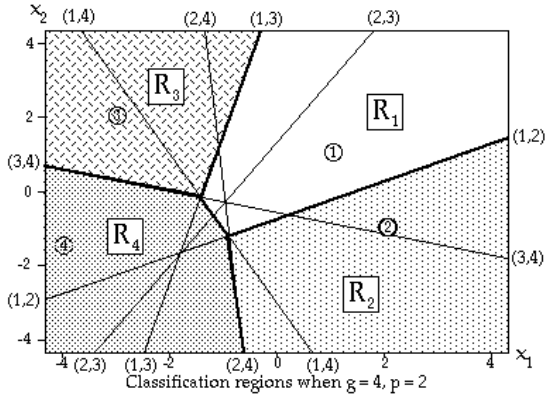
Example when $p = 2$ and $g = 4$, assuming $p_1 = p_2 = p_3 = p_4$, equal costs

The four population means are

$\mu_1 = [1, 1]'$ $\mu_2 = [2, -1]'$,

$\mu_3 = [-3, 2]'$ $\mu_4 = [-4, -1.5]'$

with variances $\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \Sigma = I_2$.



The circled numbers mark μ_1, μ_2, μ_3 and μ_4. The boundaries of pairwise preference regions are lines. Heavy lines separate $R_1,\ldots,R_4$ which are all unbounded.

A line labeled (i, j) at the boundary separates a part where π_i is preferred and a part where π_j is preferred. The equation of the line is $(\ell_i - \ell_j)'x = c_i - c_j$.

The area where a population is preferred to *all* other populations has a boundary made up of straight lines. It may be bounded or unbounded. In the plot, these lines are thick.

The is a feature of linear classification.

For this situation (all $\Sigma_i = I_2$) the discriminant function coefficients are

$$\ell_i = \Sigma^{-1}\mu_i = \mu_i,\ i = 1, 2, 3, 4$$

and the additive constants are

$$c_i = \mu_i'\Sigma^{-1}\mu_i/2 = \|\mu_i\|^2/2,\ 1 = 1,2,3,4$$

Example with Iris Data

```
Cmd> x <- read("","t11_05", quiet:T) # Iris data
Read from file "TP1:Stat5401:Data:JWData5.txt"
Cmd> variety <- factor(x[,1]); x <- x[,-1]
Cmd> stats <- tabs(x,variety,mean:T,covar:T)
Cmd> stats
component: mean   Rows are variety mean vectors
(1,1)       5.006       3.428       1.462       0.246
(2,1)       5.936       2.77        4.26        1.326
(3,1)       6.588       2.974       5.552       2.026
component: covar  3 by 4 by 4 array
(1,1,1)     0.12425     0.099216    0.016355    0.010331    S_1
(1,2,1)     0.099216    0.14369     0.011698    0.009298
(1,3,1)     0.016355    0.011698    0.030159    0.0060694
(1,4,1)     0.010331    0.009298    0.0060694   0.011106
(2,1,1)     0.26643     0.085184    0.1829      0.05578     S_2
(2,2,1)     0.085184    0.098469    0.082653    0.041204
(2,3,1)     0.1829      0.082653    0.22082     0.073102
(2,4,1)     0.05578     0.041204    0.073102    0.039106
(3,1,1)     0.40434     0.093763    0.30329     0.049094    S_3
(3,2,1)     0.093763    0.104       0.07138     0.047629
(3,3,1)     0.30329     0.07138     0.30459     0.048824
(3,4,1)     0.049094    0.047629    0.048824    0.075433
Cmd> n <- tabs(,variety); n # all three sample sizes
(1)         50          50          50
Cmd> xbar <- stats$mean' # columns are group means
Cmd> S <- stats$covar # 3 by 4 by 4 array
Cmd> spooled <- matrix(((n[1]-1)*S[1,,]+(n[2]-1)*S[2,,]+\
       (n[3]-1)*S[3,,])/sum(n-1))
```

Or compute **spooled** by

```
Cmd> manova("x = variety", silent:T)
Cmd> spooled <- matrix(SS[3,,]/DF[3])
```


White box computation.

```
Cmd> l_hat <- solve(spooled,xbar); l_hat # Coefficients
(1,1)       23.544      15.698      12.446
(2,1)       23.588      7.0725      3.6853
(3,1)      -16.431      5.2115      12.767
(4,1)      -17.398      6.4342      21.079
```

Columns are $\hat{\ell}_j = \hat{\Sigma}^{-1}\hat{\mu}_j = \hat{\Sigma}^{-1}\overline{x}_j$, $j = 1, 2, 3$.

```
Cmd> c_hat <- -sum(xbar*l_hat)/2; c_hat # row vector
              (1)         (2)         (3)
(1)       -85.21      -71.754     -103.27
```

Elements are $\hat{c}_j = -\hat{\ell}_j'\hat{\mu}_j/2$, $j = 1, 2, 3$

Scores are $\hat{\ell}_j'x + \hat{c}_j$.

Black box computation using discrim().

```
Cmd> linfun <- discrim(variety,x); linfun # find LDA stuff
   component: coefs      Columns are l_i
              variety1    variety2    variety3
   SepLen       23.544      15.698      12.446   same as l_hat
   SepWid       23.588      7.0725      3.6853
   PetLen      -16.431      5.2115      12.767
   PetWid      -17.398      6.4342      21.079
   component: addcon     row vector of additive constants
         variety1    variety2    variety3
   (1)       -85.21     -71.754     -103.27      same as c_hat

   Cmd> x_1_1 <- vector(x[1,]); x_1_1 # first case in group 1
   (1)         5.1         3.5         1.4         0.2
   Cmd> scores_1_1 <- x_1' %*% linfun$coefs + linfun$addcon;
   Cmd> scores_1_1
         variety1    variety2    variety3
   (1)       90.94       41.644      -4.8084   row vector
```

This is $[\hat{\ell}_1'x + \hat{c}_1,\ \hat{\ell}_2'x + \hat{c}_2,\ \hat{\ell}_3'x + \hat{c}_3]$

Calculate estimated posterior probabilities $e^{\hat{\ell}_j'\mathbf{x}-K(\mathbf{x})}/\sum_{1\le i\le 3}e^{\hat{\ell}_i'\mathbf{x}-K(\mathbf{x})}$, assuming equal prior probabilities:

```
Cmd> kx1 <- max(vector(scores_1_1)) # K(x)
```

kx1 is value of $K(\mathbf{x}) = \max_i(\hat{\ell}_i'\mathbf{x} + \hat{c}_i)$.

```
Cmd> exp(scores_1_1 - kx1)/sum(vector(exp(scores_1_1 - kx1)))
       variety1    variety2    variety3
(1)           1  3.8964e-22  2.6112e-42
```

It is easy to include prior probabilities.

Suppose $p_1 = .1$, $p_2 = p_3 = .45$.

```
Cmd> prior <- vector(.10, .45, .45) # Prior probabilities
Cmd> scoresP_1_1 <- log(prior)' + scores_1_1; scoresP_1_1
variety1    variety2    variety3
(1)      88.638      40.845     -5.6069   scores w/prior
Cmd> exp(scoresP_1_1)/sum(vector(exp(scoresP_1_1))) # posterior
       variety1    variety2    variety3
(1)           1  1.7534e-21   1.175e-41
```

In one step:

```
Cmd> prior'*exp(scores_1_1)/sum(vector(prior'*exp(scores_1_1)))
       variety1    variety2    variety3
(1)           1  1.7534e-21   1.175e-41
```

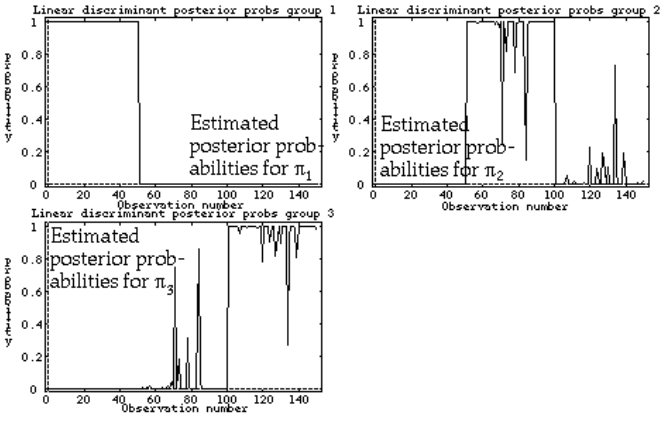
prior'*exp(scores_1_1) multiplies exp(scores_1_1)[,i] by prior[i].

Compute and plot posterior probabilities for all cases assuming equal prior probabilities:

```
Cmd> linscores <- x %*% linfun$coefs + linfun$addcon # 150 by 3
Cmd> kx <- max(linscores')' # row maxima,              150 by 1
Cmd> linprobs <- exp(linscores - kx)/sum(exp(linscores - kx)')'
Cmd> head(linprobs)
       variety1    variety2    variety3
(1)           1  3.8964e-22  2.6112e-42
(2)           1   7.218e-18  5.0421e-37
(3)           1  1.4638e-19  4.6759e-39
(4)           1  1.2685e-16  3.5666e-35
(5)           1  1.6374e-22  1.0826e-42
(6)           1  3.8833e-21  4.5665e-40
(7)           1  1.1135e-18  2.3026e-37
(8)           1  3.8776e-20  1.0745e-39
(9)           1  1.9028e-15  9.4829e-34
(10)          1  1.1118e-18  2.7241e-38
```

This seems to show it is certain (probability = 1) that these cases come from variety 1. That rests on the truth of the assumption that you know all the possible populations. Without that, it is possible they were from a different variety that was similar to but not the same as variety 1.


```
Cmd> for(i,run(3)){lineplot(1,linprobs[,i],\
 title:paste("Linear discriminant posterior probs group", i),\
 xlab:"Observation number",ylab:"Probability" )}
```



These are plots of the estimated $P(\pi_i \mid \mathbf{x})$ against case number.

For cases in group 1 (cases 1 - 50), the top left graphs shows $\hat{P}(1 \mid \mathbf{x}) \tilde{=} 1$, $\mathbf{x}$ in π_1, $\hat{P}(1 \mid \mathbf{x}) \tilde{=} 0$ $\mathbf{x}$ not in π_1

The top right graph shows there are 2 cases in π_2 with $\hat{P}(3 \mid \mathbf{x}) > .5$ and one case in π_3 with $\hat{P}(2 \mid \mathbf{x}) > .5$. The rule would misclassify these 3 cases.

Quadratic discrimination

When the variance matrices $\mathbf{\Sigma}_i$ of normal distributions differ, the minimum ECM and TMP rules are *not* linear.

The boundary lines or surfaces separating regions R_i are *curved*.

The density for population i is

$$f_i(\mathbf{x}) = \frac{\exp\{-(\mathbf{x}-\boldsymbol{\mu}_i)'\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)/2\}}{(2\pi)^{p/2}\det(\mathbf{\Sigma}_i)^{1/2}}$$

You can ignore $(2\pi)^{p/2}$, but *not* $\det(\mathbf{\Sigma}_i)^{1/2}$.

The minimum TPM rule, "select π_i to maximize $p_i f_i(\mathbf{x})$", becomes

- Select π_i so that

$\log p_i - \log(\det(\mathbf{\Sigma}_i))/2 - (\mathbf{x}-\boldsymbol{\mu}_i)'\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)/2$ is as large as possible.

$\log p_i - \log(\det(\boldsymbol{\Sigma}_i))/2 - (\mathbf{x}-\boldsymbol{\mu}_i)'\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)/2$, is a **score**, computed from the data for each π_i. With equal costs, you assign $\mathbf{x}$ to the group with the highest score.

You can rewrite this "score" as

$$\log p_i - \log(\det(\boldsymbol{\Sigma}_i))/2 - Q_i(\mathbf{x}) + L_i(\mathbf{x}) - c_i$$

where, with $\overline{\boldsymbol{\mu}} \equiv \sum \boldsymbol{\mu}_i/g$ = grand mean,

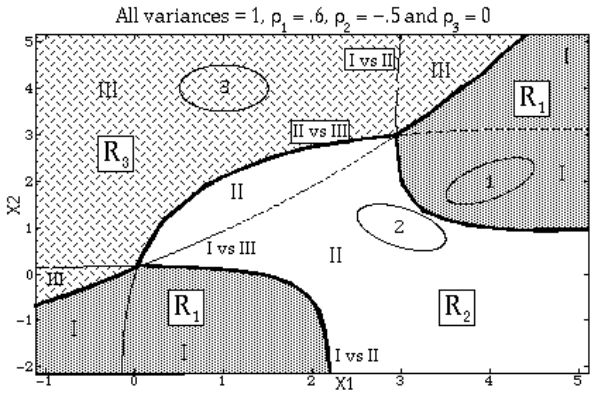
- $Q_i(\mathbf{x}) = (\mathbf{x} - \overline{\boldsymbol{\mu}})'\boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \overline{\boldsymbol{\mu}})/2$
- $L_i(\mathbf{x}) = \boldsymbol{\ell}_i'(\mathbf{x} - \overline{\boldsymbol{\mu}})$, $\boldsymbol{\ell}_i = \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})$
- $c_i = (\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})'\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})/2 = \boldsymbol{\ell}_i'(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})/2$

$Q_i(\mathbf{x})$ is a quadratic function of $\mathbf{x}$.

In this, you can replace $\overline{\boldsymbol{\mu}}$ by any convenient vector, including $\mathbf{0}$.

In place of $\overline{\boldsymbol{\mu}}$, MacAnova **discrimquad()** uses the grand mean of the training sample (ignoring groups) .

The preference regions are separated by curved boundaries. They need not be connected and may have holes. Here is a bivariate three population (p = 2, g = 3) example:



The ellipses indicate the shape of each distribution.

Note that R_1 is disconnected. This is impossible with linear discrimination.

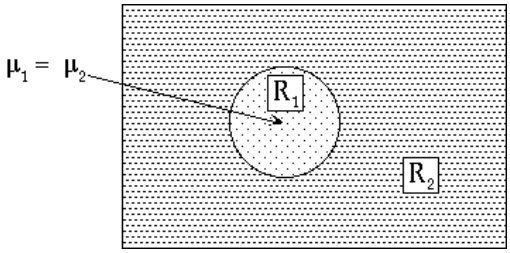
There are no holes in any R_i in this example, but holes are possible.

Here is an example with a hole.

When p = 2, g = 2, $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ but

- $\boldsymbol{\Sigma}_1 = \mathbf{I}_2$
- $\boldsymbol{\Sigma}_2 = 10\mathbf{I}_2 \neq \boldsymbol{\Sigma}_1$

the regions look like this:



The region R_2 where $\hat{\pi}(\mathbf{x}) = \pi_2$ has a *hole* in it and R_1 fills in the hole.

This says that you classify points far away from the common mean in the population with the larger variances and points near the common mean in the population with smaller variances. Linear classification would be useless in this case.

You can do quadratic classification or discrimination using **discrimquad()**.

```
Cmd> quadfun <- discrimquad(variety, x); quadfun
component: Q  Coefficients of quadratic part
  variety 1        -.5*Inverse of S1
             SepLen      SepWid      PetLen      PetWid
SepLen     -9.4717      6.2024      2.2501      2.3881
SepWid      6.2024     -7.7853    -0.55554       1.052
PetLen      2.2501    -0.55554     -19.388      8.9675
PetWid      2.3881       1.052      8.9675     -53.023
  variety 2        -.5*Inverse of S2
             SepLen      SepWid      PetLen      PetWid
SepLen     -4.7514      1.8381      4.3159     -3.2273
SepWid      1.8381     -9.8555      -1.058      9.7402
PetLen      4.3159      -1.058     -9.9019      13.469
PetWid     -3.2273      9.7402      13.469     -43.622
  variety 3        -.5*Inverse of S3
             SepLen      SepWid      PetLen      PetWid
SepLen     -5.2669      1.7399      4.9801    -0.89408
SepWid      1.7399     -7.9377    -0.55134      4.2364
PetLen      4.9801    -0.55134     -6.7029      1.4455
PetWid    -0.89408      4.2364      1.4455      -9.657
component: L      Components are coefficient vectors l_i
  variety 1       Plain (column) vector
     SepLen      SepWid      PetLen      PetWid
    -5.5743      15.613     -67.752     -56.699
  variety 2
     SepLen      SepWid      PetLen      PetWid
    -1.5787     -7.4095      5.1216       3.724
  variety 3
     SepLen      SepWid      PetLen      PetWid
    -8.2561     -8.9402      14.151      12.818
component: addcon Elements are -c_i -  .5*log(det(s_i))
      variety 1   variety 2   variety 3
(1)      -103.5      2.9244     -10.827   row vector
component: grandmean Average over all cases, column vector
      SepLen      SepWid      PetLen      PetWid
      5.8433      3.0573       3.758      1.1993
```

Component **grandmean** is $\overline{\overline{\mathbf{x}}} = (1/N)\sum_{1 \le i \le N} \mathbf{x}_i$.

Components of **L** are $\mathbf{S}_j^{-1}(\overline{\mathbf{x}}_j - \overline{\overline{\mathbf{x}}})$.

Check some of these results

```
Cmd> x1 <- x[variety == 1,] # all group 1 data

Cmd> s1 <- tabs(x1,covar:T); xbar1 <- describe(x1,mean:T)

Cmd> labs <- getlabels(x,2) # get variable labels

Cmd> setlabels(s1,structure(labs,labs)) # add them to s1

Cmd> gmean <- describe(x,mean:T); gmean # grand mean
(1)      5.8433      3.0573       3.758      1.1993

Cmd> -.5*solve(s1)    # quadfun$Q[1]
             SepLen      SepWid      PetLen      PetWid
SepLen     -9.4717      6.2024      2.2501      2.3881
SepWid      6.2024     -7.7853    -0.55554       1.052
PetLen      2.2501    -0.55554     -19.388      8.9675
PetWid      2.3881       1.052      8.9675     -53.023

Cmd> l_1 <- vector(solve(s1,xbar1 - gmean)); l_1
      SepLen      SepWid      PetLen      PetWid
     -5.5743      15.613     -67.752     -56.699   quadfun$L[1]

Cmd> -.5*l_1' %*% (xbar1 - gmean) - .5*log(det(s1))
(1,1)      -103.5                                  quadfun$addcon[1]
```

Compute quadratic scores for case 1.

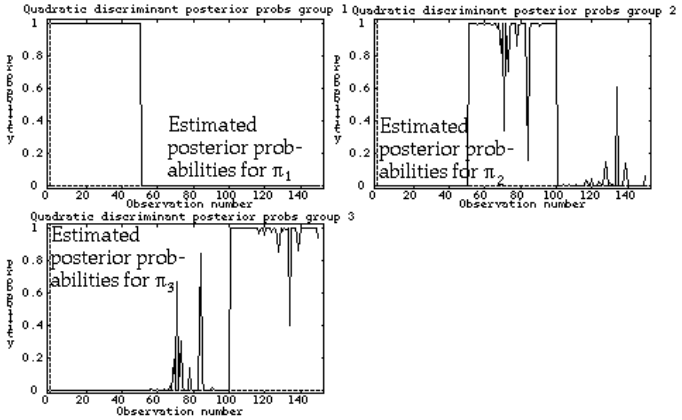
```
Cmd> x_1_1 <- x[1,]' # column vector

Cmd> qscore_1_1 <- rep(0,3) # place to put scores

Cmd> d <- x_1_1 - quadfun$grandmean
```

d is $\mathbf{x}_{11} - \hat{\boldsymbol{\mu}}$.

```
Cmd> for(i,1,3){ # compute score for each group
        qscore_1_1[i] <- d' %*% quadfun$Q[i] %*% d + \
        quadfun$L[i]' %*% d + quadfun$addcon[i]
     ;;}

Cmd> qscore_1_1 # since score[1] is max, classify in group 1
(1)      6.3091      -51.965      -87.004

Cmd> exp(qscore_1_1)/sum(exp(qscore_1_1)) # posterior probs
(1)           1  4.9185e-26  2.9815e-41

Cmd> probsquad(x_1_1,quadfun) # macro computes posterior probs
      variety 1   variety 2   variety 3
(1)           1  4.9185e-26  2.9815e-41
```


It's easy to compute posterior probabilities for every case once the quadratic discriminant function coefficients have been computed. **probsdquad()** accepts a matrix as first argument, computing a matrix of posterior probabilities, a row for each row.

```
Cmd> quadprobs <- probsquad(x,quadfun); quadprobs[1,,] #nxg
      variety 1   variety 2   variety 3
(1)           1  4.9185e-26  2.9815e-41

Cmd> for(i,run(3)){lineplot(1,quadprobs[,i],\
 title:paste("Quadratic discriminant posterior probs group",\
 i),xlab:"Observation number",ylab:"Probability")}
```



These look almost identical to the graphs of linear classification probabilities.