Displays for Statistics 5401

Lecture 37

December 5, 2005

Christopher Bingham, Instructor
612-625-1024

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

Evaluation of an estimated rule

When you find a classification rule, whether it's based on non-parametric or parametric estimation of densities from a training sample, you end up with an <u>estimated</u> rule $\hat{\hat{\pi}}(\mathbf{x})$.

The estimated <u>minimum TPM</u> rule is

- $\hat{\hat{\pi}}(\mathbf{x})$: Select π_j to maximize $p_i\hat{f}_i(\mathbf{x})$

The double "hats" signify that the rule is only an estimate of the optimal $\hat{\pi}$.

How good is $\hat{\hat{\pi}}$? That is, how small are

$$TPM = TPM(\hat{\hat{\pi}}) \equiv \sum_{1\le i\le g} p_i\{\sum_{j\ne i} P(j \mid i;\hat{\hat{\pi}})\}$$

and/or

$$ECM = ECM(\hat{\hat{\pi}}) = \sum_{1\le i\le g} p_i\{\sum_{j\ne i} P(j \mid i;\hat{\hat{\pi}})C(j \mid i)\}$$

Both TPM and ECM depend on $\{p_i\}$ and $P(j \mid i;\, \hat{\hat{\pi}})$, $i \ne j$.

It is presumed the $\{C(j \mid i)\}$ are known quantities.

Usually <u>prior probabilities</u> p_i are also treated as known.

Sometimes you can estimate the p_i from **sample proportions** $\hat{p}_i = n_i/N$, n_i = number of cases from π_i in the training sample. This is possible when you obtain the training sample by "<u>mixture sampling</u>"

- randomly select π_i with probability p_i,
- then select of $\mathbf{x}$ using density $f_i(\mathbf{x})$)

But you still need estimates of misclassification probabilities $P(j \mid i;\hat{\hat{\pi}})$ to estimate either $TPM(\hat{\hat{\pi}})$ or $ECM(\hat{\hat{\pi}})$.

$P(j \mid i;\hat{\hat{\pi}})$ depends on the densities $f_i(\mathbf{x})$ and is ***not*** known.

There are two <u>different</u> questions that might be of interest:

$\mathbf{Q}_1$ How do you estimate $P(j \mid i;\hat{\pi})$ = classification probabilities for the <u>actual</u> best $\hat{\pi}$?

$\mathbf{Q}_2$ How do you estimate $P(j \mid i;\hat{\hat{\pi}})$ = classification probabilities for $\hat{\hat{\pi}}$ the <u>estimated</u> best $\hat{\pi}$?

An answer to Q_1 might interest a mathematical statistician.

An answer to Q_2 is more useful because, in practice, you will use $\hat{\hat{\pi}}$ to classify future cases, not $\hat{\pi}$.

In parametric situations (densities of known form with parameter vectors $\boldsymbol{\theta}_i$),

$$P(i|j;\hat{\hat{\pi}}) = g_{ij}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j),$$

where g_{ij} depends of the $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ that characterize π_i and π_j.

When you can find a <u>mathematical formula</u> for $g_{ij}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = P(i|j;\hat{\hat{\pi}})$ and can estimate $\boldsymbol{\theta}_i$ by $\hat{\boldsymbol{\theta}}_i$, i = 1, ..., g, you can estimate $P(i|j;\hat{\hat{\pi}})$ consistently by

$$\hat{P}(i|j;\hat{\hat{\pi}}) = g_{ij}(\hat{\boldsymbol{\theta}}_i,\hat{\boldsymbol{\theta}}_j), \text{ all } i, j.$$

Example of formula

- $g = 2$, $p = 2$
- $\mathbf{x}$ either $N_2(\boldsymbol{\mu}_1,\boldsymbol{\Sigma})$ (π_1) or $N_2(\boldsymbol{\mu}_2,\boldsymbol{\Sigma})$ (π_2).
- prior probabilities $p_1 = p_2 = .5$

Then

$$P(1|2) = P(2|1) = g_{12}(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\boldsymbol{\Sigma}) = g_{21}(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\boldsymbol{\Sigma}) = \Phi(-\Delta/2),$$

$$\Phi(x) = \textbf{cumnor}(x) = \frac{1}{\sqrt{\{2\pi\}}}\int_{-\infty}^{x} e^{-z^2/2}dz$$

$$\Delta^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

Therefore,

$$TPM = .5*P(1|2) + .5*P(2|1)) = \Phi(-\Delta/2)$$

You estimate TPM and $P(i|j)$ by

$$\widehat{TPM} = \hat{P}(1|2) = \hat{P}(2|1) = \Phi(-\hat{\Delta}/2)$$

where

$$\hat{\Delta} = \sqrt{\{(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)'\mathbf{S}^{-1}(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2)\}},\ \mathbf{S} = \mathbf{S}_{pooled}$$

When you don't have a formula for $g_{ij}(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j)$, you may be able to estimate $g_{ij}(\hat{\boldsymbol{\theta}}_i,\hat{\boldsymbol{\theta}}_j)$ by ***simulation***:

1. Generate large samples from densities $f_j(\mathbf{x};\hat{\boldsymbol{\theta}}_j)$
2. Use $\hat{\hat{\pi}}$ to classify them;
3. estimate $\hat{P}(i|j;\hat{\hat{\pi}}) = g_{ij}(\hat{\boldsymbol{\theta}}_i,\hat{\boldsymbol{\theta}}_j)$ by <u>relative frequencies</u>.

Warning: $g_{ij}(\hat{\boldsymbol{\theta}}_i,\hat{\boldsymbol{\theta}}_j)$ is useful only when the parametric model is <u>correct</u>. Even when you assume multivariate normality to *derive* $\hat{\hat{\pi}}$, you probably shouldn't believe it accurately describes the populations.

It is better to have a way to estimate $P(i|j;\hat{\hat{\pi}})$ regardless of the model, that is, see how well it classifies <u>actual data</u> from <u>known populations</u>.

Best situation (validation sample): You have two *independent* data sets

- the <u>training sample</u> $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_N$ which you use to find rule $\hat{\hat{\pi}}(\mathbf{x})$
- a <u>validation sample</u> $\mathbf{x}_1^*,\mathbf{x}_2^*,\mathbf{x}_3^*,\ldots,\mathbf{x}_M^*$ which is <u>independent</u> which you use to estimate $TPM(\hat{\hat{\pi}})$ or $ECM(\hat{\hat{\pi}})$.

In both samples you know the population each $\mathbf{x}_i$ or $\mathbf{x}_i^*$ belongs to.

Suppose there are M_j cases from π_j in the validation sample.

- Use $\hat{\hat{\pi}}(\mathbf{x}_k^*)$ to classify case k in the *validation* sample, k = 1,...,M.
- Find m_{ij} = number of $\mathbf{x}^*$ values from π_j with $\hat{\hat{\pi}}(\mathbf{x}^*) = \pi_i$
- Then $\hat{P}(i\,|\,j;\hat{\hat{\pi}}) = m_{ij}/M_j$ = *sample proportion*.

Because $E[m_{ij}] = M_j \times P(i\,|\,j;\hat{\hat{\pi}})$, $\hat{P}$ is unbiased, that is

$$E[\hat{P}(i\,|\,j;\,\hat{\hat{\pi}})] = P(i\,|\,j;\,\hat{\hat{\pi}}).$$

The estimated TPM and ECM are

$$\widehat{TPM} = \sum_{1\le j\le g} p_j\{\sum_{i\ne j} m_{ij}/M_j\}$$
$$= 1 - \sum_{1\le j\le g} p_j m_{jj}/M_j$$
$$\widehat{ECM} = \sum_{1\le j\le g} p_j\{\sum_{i\ne j} C(i\,|\,j)m_{ij}/M_j\}$$

Because $\hat{P}(i\,|\,j;\,\hat{\hat{\pi}})$ is unbiased, $\widehat{TPM}$ and $\widehat{ECM}$ are also unbiased.

Disadvantage:

- You could probably get a better estimate of the optimal rule using *both* the training and validation samples to estimate the rule.

This is not an important issue when the training sample is sufficiently large to provide an acceptable classification rule.

Re-use the training sample

What if you were to re-use the training sample as if it were a validation sample?

That is, use $\hat{\hat{\pi}}$ to classify cases in the training sample and determine the relative frequencies of errors.

This will have optimistic bias, that is, it will make $\hat{\hat{\pi}}$ appear to be too good.

- Let n_{ij} = number of $\mathbf{x}$'s in the training sample that belong in π_j' but $\hat{\hat{\pi}}(\mathbf{x}) = \pi_i$. Then the estimate based on the training sample is $\hat{P}(i\,|\,j;\,\hat{\hat{\pi}}) \equiv n_{ij}/N_j$.

Estimated TPM and ECM are

- $\widehat{TPM} = \sum_{1\le j\le g} p_j\{\sum_{i\ne j} n_{ij}/N_j\} = 1 - \sum_{1\le j\le g} p_j n_{jj}/N_j$
- $\widehat{ECM} = \sum_{1\le j\le g} p_j\{\sum_{i\ne j} C(i\,|\,j)n_{ij}/N_j\}$

But $E[\widehat{TPM}] < TPM$ and $E[\widehat{ECM}] < ECM$.

If you estimate p_j by $\hat{p}_j = N_j/N$, $\widehat{TPM}$ is

$$\widehat{TPM} = \sum_{j\ne i} n_{ij}/N$$
$$= (\text{total number of errors})/N$$

This is the **APER** = *apparent error rate.*

APER is directly applicable only when $\hat{p}_j = N_j/N$ estimates p_j. Even in that case, the APER is biased *downward* and is thus "optimistic".

$\hat{p}_j = N_j/N$ is a sensible estimate of the prior probabilities p_j only when

- $\{p_j\}$ are objective probabilities describing the prevalence of the populations,
- The training sample was collected randomly, with the probability of each case belonging to population π_j being p_j.

Leave-one-out (Jackknife method)

This widely used method <u>reuses the training sample</u> to estimate <u>classification probabilities</u> of an estimated rule. It is also known as ***Lachenbruch's holdout*** procedure or the ***Lachenbruch-Mickey*** method.

You classify each case in the training sample using a rule based on all the <u>other</u> cases, "holding out" the data for the case you are classifying. Specifically:

- For case k, k = 1, ..., N, estimate a rule $\hat{\hat{\pi}}_{(-k)}(\mathbf{x})$ using all the data except $\mathbf{x}_k$.
- Then classify each $\mathbf{x}_k$ as coming from population $\hat{\hat{\pi}}_{(-k)}(\mathbf{x}_k)$.
- Estimate $P(i\,|\,j)$ by $\hat{P}(i\,|\,j) = n_{ij}^*/N_j$ where n_{ij}^* = number of π_j cases classified as π_i.

$$\widehat{TPM}_{JK} = \sum_{1\le j\le g} p_j \sum_{i\ne j} \hat{P}(i\,|\,j) = \sum_{1\le j\le g} p_j \left(\sum_{i\ne j} n_{ij}^*\right)/N_j$$

When you estimate p_j by $\hat{p}_j = N_j/N$,

$$\widehat{TPM}_{JK} = \left(\sum_{1\le j\le g}\sum_{i\ne j} n_{ij}^*\right)/N = 1 - \sum_{1\le j\le g} n_{jj}^*/N$$

This might be called $APER_{JK}$ = <u>JackKnife Apparent Error Rate</u>. It estimates TPM only when you obtained the training sample by "mixture" sampling.

Example

```
Cmd> data <- read("","dayfisher")
dayfisher     29      8 format
) Data on seed of Pantago maritima (a Scottish plantain) was
) collected from 29 Scottish localities.  The localities could
) be broadly classified in three regions - Coastal, Inland and
) Island.  The Costal localities were further subdivided into
) 4 types (a) waterlogged mud, (b) typical salt marsh, (c)
) drained mud) and (d) coastal meadow above highest tide mark.
)
) Seed from each locality was grown under comparable
) conditions in an experimental garden and various
) measurements made on 100 plants grown from seed from each
) locality.  In particular sample means and standard
) deviations were computed for leaf length (L), breadth (B)
) and thickness (T).
)
) This data consists of 1000*log10(means) and 1000*log10(stdev)
) Col. 1: Region (1 = Coastal, 2 = Inland, 3 = Island)
) Col. 2: Locality within Coastal region 1 (1=type a,2=type b,
)         3=type c,4=type d); also 1 for all of regions 2 and 3
) Col. 3: 1000*log10(stdev(length))
) Col. 4: 1000*log10(stdev(breadth))
) Col. 5: 1000*log10(stdev(thickness))
) Col. 6: 1000*log10(mean(length))
) Col. 7: 1000*log10(mean(breadth))
) Col. 8: 1000*log10(mean(thickness))
Read from file "TP1:Stat5401:dayfisher.mat"
```

```
Cmd> region <- factor(data[,1])

Cmd> locality <- factor(data[,2])

Cmd> y <- data[,-run(2)]
```

From the description in the header, **locality** is nested within **region**. I combined these in a single factor **place** with 6 levels.

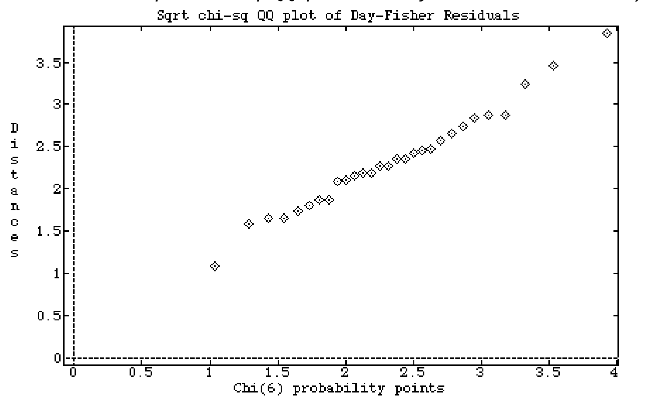
```
Cmd> unique(10*region + locality) # 6 different values
(1)         11         12         13         14         21         31

Cmd> place <- makefactor(10*region + locality)

Cmd> list(place)
place           REAL   29    1     FACTOR with 6 levels
```

This works because **10*region+locality** consists of the 6 numbers - 11, 12, 13, 14, 21, and 31 which **makefactor()** turns into a factor with 6 levels.

```
Cmd> manova("y = place")# do MANOVA computates to get residuals
Model used is y = place
WARNING: summaries are sequential
NOTE: SS/SP matrices suppressed because of size; use
'manova(,sssp:T)'
                              SS and SP Matrices
                   DF
CONSTANT            1
                        Type 'SS[1,,]' to see SS/SP matrix
place               5
                        Type 'SS[2,,]' to see SS/SP matrix
ERROR1             23
                        Type 'SS[3,,]' to see SS/SP matrix
```

Check that things aren't too far from normal.

```
Cmd> chiqqplot(RESIDUALS,sqrt:T,\
            title:"Sqrt chi-sq QQ plot of Day-Fisher Residuals")
```



It looks pretty straight.

```
Cmd> N <- nrows(y) # size of training sample (29)

Cmd> setoptions(format:"9.5g") # fit more on line

Cmd> discrimfun <- discrim(place,y) # find linear discrim fun

Cmd> discrimfun # coefficients in estimated min TPM rule
component: coefs      Coefficients of linear part
       place1     place2     place3     place4     place5     place6
(1)   0.83379    0.84765     0.8636    0.88501    0.81145    0.84027
(2)  -0.24068   -0.24764   -0.27343   -0.27526   -0.27973   -0.27775
(3)    -0.449   -0.46187   -0.47554   -0.50837   -0.45276   -0.47627
(4)   -1.0685    -1.0682    -1.0908    -1.1082    -0.9932    -1.0706
(5)    1.0617     1.0838     1.1484     1.1717     1.0317     1.0769
(6)    1.3695     1.3815     1.4215     1.4205      1.283     1.3668
component: addcon constant to be added
       place1     place2     place3     place4     place5     place6
(1)     -1145    -1178.9    -1270.1    -1293.7    -1035.1    -1142.5
```

```
Cmd> scores <- y %*% discrimfun$coefs + discrimfun$addcon

Cmd> list(scores)
scores          REAL   29    6     (labels)
```

Each row of **scores** contains the linear discriminant scores for the 6 places for that case.

Assuming equal prior probabilities $p_1 = p_2 = \ldots = p_6 = 1/6$, you classify a case in the population with the largest score.

```
Cmd> scores_1 <- vector(scores[1,]); scores_1 # case 1 scores
(1)    1103.7    1100.5    1086.3    1079.3    1095.5    1100.5
```

You would classify case 1 as place 1 since place 1 has the highest score.

Now find estimated posterior probabilities. Start with case 1.

```
Cmd> exp(scores_1)/sum(exp(scores_1))
WARNING: exp(x) with result too large set to MISSING
WARNING: exp(x) with result too large set to MISSING
WARNING: MISSING values found by sum()
WARNING: arithmetic with missing value(s); operation is /
(1)   MISSING   MISSING   MISSING   MISSING   MISSING   MISSING
```

Ooops! This is a case where I should have subtracted some function K(**x**) from the scores.

```
Cmd> kx <- scores[,1]  # scores for place 1

Cmd> probs <- exp(scores - kx)/vector(sum(exp(scores - kx)'))

Cmd> probs[1,] # repeat of posterior probs for case 1
      place1    place2    place3    place4    place5     place6
(1)  0.92492  0.036058 2.6112e-08 2.301e-11 0.00026554  0.038755
```

You would classify case 1 in crop 1 when $\hat{\pi}(\mathbf{x})$ computed from all the data.

Now I find $\hat{\hat{\pi}}(\mathbf{x}_i)$ for all cases at once.

```
Cmd> daplace <- vector(grade(probs',down:T)[1,])
```

daplace contains guessed classifications of all 29 training sample cases.

```
Cmd> print(paste(daplace))
1 1 1 2 2 2 2 2 2 2 3 3 4 4 4 4 5 5 5 5 5 5 2 5 6 6 6 5 6
```

Posterior probabilities and guesses:

```
Cmd> print(format:"5.3f",hconcat(probs, place, daplace))
MATRIX::            Probabilities                   True  Guess
  (1,1) 0.925 0.036 0.000 0.000 0.000 0.039 1.000 1.000
  (2,1) 0.893 0.107 0.000 0.000 0.000 0.000 1.000 1.000
  (3,1) 0.826 0.060 0.000 0.000 0.000 0.113 1.000 1.000
  (4,1) 0.253 0.396 0.001 0.000 0.005 0.345 1.000 2.000 *
  (5,1) 0.354 0.633 0.000 0.000 0.000 0.012 1.000 2.000 *
  (6,1) 0.286 0.700 0.001 0.000 0.000 0.013 2.000 2.000
  (7,1) 0.268 0.730 0.001 0.000 0.000 0.000 2.000 2.000
  (8,1) 0.219 0.471 0.001 0.000 0.005 0.303 2.000 2.000
  (9,1) 0.056 0.808 0.119 0.005 0.000 0.012 2.000 2.000
 (10,1) 0.082 0.417 0.005 0.000 0.391 0.104 2.000 2.000
 (11,1) 0.004 0.059 0.937 0.000 0.000 0.000 3.000 3.000
 (12,1) 0.000 0.021 0.968 0.011 0.000 0.000 3.000 3.000
 (13,1) 0.000 0.000 0.131 0.869 0.000 0.000 3.000 4.000 *
 (14,1) 0.000 0.000 0.032 0.968 0.000 0.000 4.000 4.000
 (15,1) 0.000 0.000 0.020 0.980 0.000 0.000 4.000 4.000
 (16,1) 0.000 0.000 0.022 0.978 0.000 0.000 4.000 4.000
 (17,1) 0.001 0.000 0.000 0.000 0.992 0.007 5.000 5.000
 (18,1) 0.013 0.021 0.000 0.000 0.676 0.291 5.000 5.000
 (19,1) 0.013 0.005 0.000 0.000 0.956 0.026 5.000 5.000
 (20,1) 0.000 0.000 0.000 0.000 0.990 0.010 5.000 5.000
 (21,1) 0.000 0.000 0.000 0.000 1.000 0.000 5.000 5.000
 (22,1) 0.000 0.000 0.000 0.000 0.999 0.001 5.000 5.000
 (23,1) 0.040 0.472 0.009 0.000 0.396 0.083 5.000 2.000 *
 (24,1) 0.000 0.000 0.000 0.000 0.993 0.007 5.000 5.000
 (25,1) 0.004 0.000 0.000 0.000 0.000 0.996 6.000 6.000
 (26,1) 0.145 0.074 0.000 0.000 0.103 0.679 6.000 6.000
 (27,1) 0.074 0.126 0.001 0.000 0.001 0.798 6.000 6.000
 (28,1) 0.009 0.007 0.000 0.000 0.831 0.153 6.000 5.000 *
 (29,1) 0.012 0.168 0.008 0.006 0.016 0.790 6.000 6.000
```

* means place ≠ guessed place

```
Cmd> sum(daplace != place)
(1,1)          5 misclassified

Cmd> 5/N
(1)   0.17241   (17% APER)
```

The APER = 5/29 = 17.2%. This estimates TPM only when places were selected by "mixture sampling" so that the prior probabilities can be estimated by $\hat{p}_i = N_i/N$. This is unlikely to be true.

```
Cmd> print(tabs(,place,daplace),format:"6.0f")
MATRIX:
(1,1)      3      2      0      0      0      0
(2,1)      0      5      0      0      0      0
(3,1)      0      0      2      1      0      0
(4,1)      0      0      0      3      0      0
(5,1)      0      1      0      0      7      0
(6,1)      0      0      0      0      1      4
```

This is a "confusion matrix, a cross tabulation of the data with

- Rows corresponds to the actual locations cases came from.
- Columns correspond to the guesses $\hat{\hat{\pi}}(\mathbf{x}_i)$ made for cases.

The diagonal elements are the numbers correctly clasified.

Assume equal prior probabilities ($p_1 = \ldots = p_6 = 1/6$),

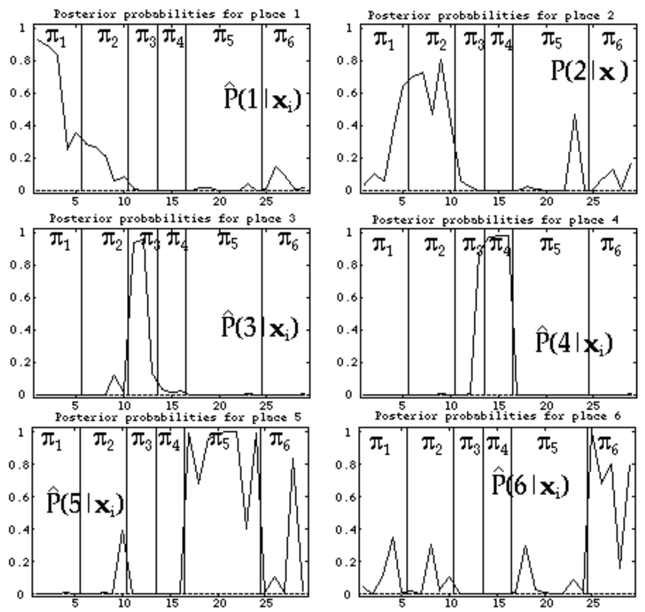
```
  Cmd> prior <- rep(1/6,6) # assumed prior probabilities

  Cmd> n <- tabs(,place); n # sample sizes
  (1)         5         5         3         3         8         5

  Cmd> P_ii <- diag(tabs(,place,daplace))/n; P_ii
  (1)       0.6         1   0.66667         1     0.875       0.8
```

These are $\hat{P}(i \mid i)$.

```
  Cmd> 1 - sum(prior*P_ii)
  (1)   0.17639
```

This is $\widehat{TPM}$, biased downward because it is computed by treating the training sample as a validation sample.

Plots against case number of the naively estimated posterior probabilities



You can tell from this that $\hat{\hat{\pi}}$ does a reasonably good job of classifying the training sample. It would probably not do as well with a true validation sample.

Jackknife estimation of error probabilities

Illustrate with case 1.

```
  Cmd> temp <- discrim(place[-1],y[-1,])
```

temp contains constants for rule computed from all the data except case 1.

```
  Cmd> scores_1 <- y[1,] %*% temp$coefs + temp$addcon

  Cmd> scores_1 <- vector(scores_1'); scores_1 # case 1 scores
     Group 1   Group 2   Group 3   Group 4   Group 5   Group 6
        1100    1097.4    1081.8    1075.5    1093.2    1097.9
```

These are the linear scores for case 1.

Compute estimated posterior probabilities $\hat{P}(j \mid \mathbf{x}_1)$ for case 1.

```
  Cmd> kx1 <- max(scores_1);exp(scores_1-kx1)/sum(exp(scores_1-
  kx1))
     Group 1   Group 2   Group 3   Group 4   Group 5   Group 6
     0.83416  0.061132 9.9361e-09 1.8822e-11 0.00088082   0.10383
```

This classifies case 1 as coming from π_1 as before. Because the maximum $\hat{P}(j \mid \mathbf{x}_1)$ is $\hat{P}(1 \mid \mathbf{x}_1) = 0.834$, you would classify with reasonable but not great confidence. Note that 0.834 < 0.925 = "naive" estimate of $P(1 \mid \mathbf{x}_1)$.

jackknife(groups,x) automates leave-one-out computations for linear classification.

```
  Cmd> pjk <- jackknife(place, y)

  Cmd> list(pjk)#Cols 1-6 are posterior probs
  pjk             REAL   29    7
```

The number of columns is g + 1 = 6 + 1

- **pjk[i,j]** contains $\hat{P}(j \mid \mathbf{x}_i)$ for case i, j = 1,..., g = 6
- **pjk[i,g+1] = pjk[i,7]** holds the group number j_{max} with
  $$\max_j \hat{P}(j \mid \mathbf{x}_i) = \hat{P}(j_{max} \mid \mathbf{x}_i).$$
  Case i would be classified as coming from $\pi_{j_{max}}$.

```
Cmd> print(format:"5.3f",hconcat(p,place))
MATRIX:     Posterior Probabilities      Guess  True
 (1,1) 0.834 0.061 0.000 0.000 0.001 0.104 1.000 1.000
 (2,1) 0.054 0.909 0.037 0.000 0.000 0.000 2.000 1.000 *
 (3,1) 0.539 0.107 0.000 0.000 0.000 0.353 1.000 1.000
 (4,1) 0.059 0.467 0.001 0.000 0.009 0.464 2.000 1.000 *
 (5,1) 0.054 0.932 0.001 0.000 0.001 0.012 2.000 1.000 *
 (6,1) 0.523 0.453 0.001 0.000 0.000 0.023 1.000 2.000
 (7,1) 0.684 0.313 0.003 0.000 0.000 0.000 1.000 2.000
 (8,1) 0.287 0.296 0.002 0.000 0.009 0.406 6.000 2.000 *
 (9,1) 0.084 0.602 0.280 0.011 0.000 0.022 2.000 2.000
(10,1) 0.049 0.026 0.006 0.000 0.773 0.146 5.000 2.000 *
(11,1) 0.166 0.829 0.002 0.000 0.000 0.003 2.000 3.000 *
(12,1) 0.001 0.143 0.799 0.057 0.000 0.000 3.000 3.000
(13,1) 0.000 0.000 0.000 1.000 0.000 0.000 4.000 3.000 *
(14,1) 0.000 0.000 0.162 0.838 0.000 0.000 4.000 4.000
(15,1) 0.000 0.007 0.505 0.488 0.000 0.000 3.000 4.000 *
(16,1) 0.000 0.000 0.047 0.953 0.000 0.000 4.000 4.000
(17,1) 0.004 0.001 0.000 0.000 0.976 0.020 5.000 5.000
(18,1) 0.029 0.047 0.000 0.000 0.348 0.576 6.000 5.000 *
(19,1) 0.183 0.052 0.000 0.000 0.630 0.136 5.000 5.000
(20,1) 0.000 0.000 0.000 0.000 0.956 0.044 5.000 5.000
(21,1) 0.000 0.000 0.000 0.000 1.000 0.000 5.000 5.000
(22,1) 0.000 0.000 0.000 0.000 0.998 0.002 5.000 5.000
(23,1) 0.077 0.787 0.005 0.000 0.035 0.097 2.000 5.000 *
(24,1) 0.000 0.000 0.000 0.000 0.984 0.016 5.000 5.000
(25,1) 0.437 0.000 0.000 0.000 0.000 0.563 6.000 6.000
(26,1) 0.323 0.157 0.000 0.000 0.258 0.262 1.000 6.000 *
(27,1) 0.139 0.240 0.001 0.000 0.002 0.618 6.000 6.000
(28,1) 0.001 0.001 0.000 0.000 0.999 0.000 5.000 6.000 *
(29,1) 0.022 0.465 0.019 0.006 0.040 0.447 2.000 6.000 *

Cmd> confus <- tabs(,place,pjk[,7]);print(confus,format:"4.0f")
confus:
(1,1)    2    3    0    0    0    0
(2,1)    2    1    0    0    1    1
(3,1)    0    1    1    1    0    0
(4,1)    0    0    1    2    0    0
(5,1)    0    1    0    0    6    1
(6,1)    1    1    0    0    1    2
```

```
Cmd> N - sum(diag(confus))
(1)        15 Number of misclassifications
```

APER_{JK} is 15/29 = .517

```
Cmd> 1 - sum(prior*diag(confus)/n)
(1)   0.54167.
```

Assuming equal p_j,

$\widehat{\text{TPM}}_{JK} = 0.54167$

As you might expect, these are larger than the naive estimates. Usually there is not that big a difference, but here no sample sizes is large ($\max\{N_j\} = 10$) and there are two samples of size 3.

Incidentally, you cannot estimate quad-ratic discriminant functions from these data since $N_j \leq p$ for several j. You need $\mathbf{S}_j^{-1}$ to compute the quadratic discriminant function for group j, but $\mathbf{S}_j$ is singular and has no inverse when $N_j \leq p = 7$ as is the case for every place except place 5.