Part 8: Linear Classification

Mark Bebbington
AgHortC 2.09A
m.bebbington@massey.ac.nz

Reading for this Part

There is very little in the text on this part (Sections 10.1, 10.2).

Copies of relevant material from Hastie, T., Tibshirani, R., Friedman, J. “The Elements of Statistical Learning” can be found on the webpage.

Linear Methods for Classification

- Assume that the output takes on a finite number of `classes’, which we will denote by G $\in$ 1,2,…,K
- One way of predicting this is to fit a linear regression model $\hat{y}_k = \hat{\beta}_{k0} + \hat{\beta}_k^T x$ to the kth indicator response variable (see later)
- We then assign the prediction to the closest class
  - The decision boundary between classes k and l is the set of points for which $\hat{y}_k = \hat{y}_l$, i.e, the set $\{x : (\hat{\beta}_{k0} - \hat{\beta}_{l0}) + (\hat{\beta}_k - \hat{\beta}_l)^T x = 0\}$, an affine set, or hyperplane. Hence `separating hyperplanes’.

Nearest Neighbour Methods

Output of an input should be similar to outputs from other inputs `close’ in input space

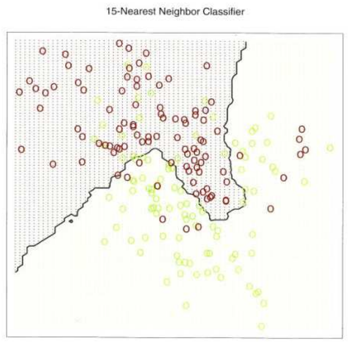

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Linear Methods

Fit straight `line'

(hyperplane) through data

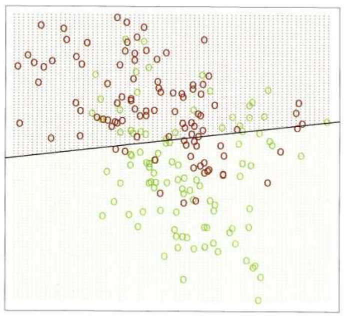

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable—*GREEN $=0$, RED $=1$*—and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta}=0.5$. *The red shaded region denotes that part of input space classified as* RED*, while the green region is classified as* GREEN.

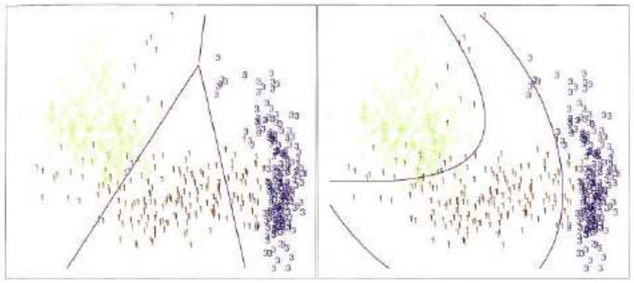

FIGURE 4.1. *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_1X_2, X_1^2, X_2^2$. *Linear inequalities in this space are quadratic inequalities in the original space.*

Vowel Data

The vowel training data has 11 classes and 10 predictors.

Difficult classification problem – best methods have approx. 40% error rates (on train/test data).

Vowel Training Data

```
data = loadtxt('vowel_train.txt')
p = 10
data = data.reshape(-1,p+1) # 1 class and 10 predictors

## reformat data as X and Y
y = data[:,0]
N = len(y)
y = transpose(matrix(y))
X = data[:,1:p+1]
X = matrix(X)

print y

[[  1.]
 [  2.]
 [  3.]
   ...
 [  9.]
 [ 10.]
 [ 11.]
```

Linear regression of an indicator matrix

- The response categories k=1,…,K are coded via an indicator variable, Y_k, where $Y_k=1$ if G=k, 0 otherwise.
- Collect these in a vector $Y = (Y_1,\ldots,Y_K)$, so that the n training instances form an N by K indicator response matrix Y of 0's and 1's, with each row having a single 1 (the class of that target instance.
- A linear regression model is fit to all the columns simultaneously

$$\hat{Y} = X(X^TX)^{-1}X^TY = X\hat{B}Y$$

Indicator Matrix in NumPy

```
# make an indicator matrix Y
K = 11 # number of classes
Y = zeros((N,K))
for n in range(N):
    Y[n,int(y[n])-1] = 1
print Y

[[ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  1.  0. ...,  0.  0.  0.]
 [ 0.  0.  1. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  1.  0.  0.]
 [ 0.  0.  0. ...,  0.  1.  0.]
 [ 0.  0.  0. ...,  0.  0.  1.]]
```


Classification

- A new observation with input x is classified as follows:
  - Compute the fitted output

$$\hat{y}(x) = [(1, x)\hat{B}]^T$$

  - Identify the largest component and classify accordingly:

$$\hat{G} = \arg\max_k \hat{y}_k(x)$$

Linear regression of an indicator matrix in NumPy

```
## Predictors in X, training classes in y,
## number of classes in K, N = len(y)
## Indicator matrix of classes in Y

Yhat = linregp(X,Y)[1]
YhatT = transpose(Yhat)  ## because the next command
                         ## works on columns ...
yhatT = YhatT.argmax(0) + 1  ## get the index of the row which has
                             ## the maximum value in each column,
                             ## add 1 to get the class
                             ## (indexing starts at `0')
yhat = transpose(yhatT)  ## convert back to a column containing the
                         ## estimated class
errorrate = (0. + N - sum(y == yhat))/N   ## N minus the number
                                          ## correct divided by N
print 'errorrate = ',errorrate

errorrate =  0.477272727273
```

- Program in 'vowel_indreg.py'

Why an indicator matrix?

Since a squared norm is a sum of squares, the components decouple and can be rearranged as a separate linear model for each element.

This is only so because there is nothing in the model which binds the different responses together

Exercise 8.1

Check the error rate of the indicator regression (as fitted to the training data) on the vowel test data, and compare with the result of fitting the model to the test data.

Linear Discriminant Analysis

- The regression approach is a discriminant function.
  - These form a discriminant function $\delta_k(x)$ for each class k, and then classify x to the class with the largest discriminant value for that x.
- The decision boundary is linear in x, if some monotone transformation of $\delta_k(x)$ or Pr(G=k |X=x) is linear in x

Example: logit transformation

If we have two classes, let

$$\Pr(G = 1 \mid X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T x)}$$

$$\Pr(G = 2 \mid X = x) = \frac{1}{1 + \exp(\beta_0 + \beta^T x)}$$

which is the logit transformation log(p/(1-p)). The decision boundary is the set of points for which the log-odds are zero, which is a hyperplane defined by $\{x|\beta_0+\beta^T x=0\}$.

Discriminant Functions

We will look at two different methods:

- linear discriminant analysis (LDA)
- linear logistic regression
- The main difference is in how we fit the training data.

Bayes Theorem

We want to obtain posterior probabilities Pr(G|X), i.e., the probability that a point is in class G, given it has value X. Recall Bayes Theorem

$$\Pr(G|X)=\Pr(X|G)\Pr(G)/\Pr(X)$$

where

$$\Pr(X)=\Sigma_g \Pr(X|g)\Pr(g)$$

(Theorem of Total Probability). So we need a *prior probability* Pr(G), and a *conditional probability* Pr(X|G).

Linear discriminant analysis

Suppose that the prior probability of being in class k is π_k, where $\Sigma_k \pi_k = 1$.

And then that each class density (~Pr(X|g)) is a multivariate Gaussian

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right)$$

In other words, that each class has a mean value μ_k, and a covariance matrix Σ_k.

Linear discriminant analysis is the special case $\Sigma_k = \Sigma$ for all k.

Linear discriminant analysis

To compare two classes k and l, the log-ratio of the probabilities is, from Bayes Theorem,

$$\log\frac{\Pr(G=k\mid X=x)}{\Pr(G=l\mid X=x)} = \log\frac{f_k(x)}{f_l(x)} + \log\frac{\pi_k}{\pi_l}$$
$$= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k-\mu_l) + x^T\Sigma^{-1}(\mu_k-\mu_l)$$

an equation linear in x. Thus the decision boundary between k and l (where the log-ratio = 0) is linear (in p dimensions, a hyperplane)

Linear discriminant function

An equivalent decision rule is

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k$$

where G(x) = $\text{argmax}_k\, \delta_k(x)$. However, we do not know the parameters of the Gaussian distn, and have to estimate them from our training data:

- $\hat{\pi}_k = N_k / N$, where N_k is the number of class k obs.
- $\hat{\mu}_k = \sum_{g_i=k} x_i / N_k$
- $\hat{\Sigma} = \sum_{k=1}^{K}\sum_{g_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (N-K)$

Debugging Exercise

A program for LDA in NumPy can be found in `vowel_lda_0.py'.

a) Make it run.

b) Make it give a sensible answer for the error rate (ie, between 0 and 1).

c) Try not to look at the following slides first.

LDA in NumPy

```
nk = zeros((K,1))
for k in range(K):
    nk[k] = sum(y == k+1)
pi = (0. + nk)/N
print "pi= ",transpose(pi)
```

➢ pi= [[0.09090909 0.09090909 ... 0.09090909 0.09090909]]

```
mu = zeros((K,X.shape[1]))
for k in range(K):
    for j in range(N):
        mu[k:k+1,:] = mu[k:k+1,:] + (y[j] ==
  k+1)*X[j:j+1,:]/nk[k]
print "mu= ",mu
```

➢ mu= [[-3.3595625 0.0629375 -0.2940625 1.20333333 0.38747917 1.22189583 0.096375 0.03710417 -0.62435417 -0.161625]
...
[-2.99039583 1.463875 -0.5098125 0.37164583 -0.38039583 0.72504167 -0.08339583 0.50766667 -0.3275 -0.22672917]]

LDA in NumPy(2)

```
Sigma = zeros((X.shape[1],X.shape[1]))
for k in range(K):
    for j in range(N):
        temp = X[j:j+1,:] - mu[k:k+1,:]
        Sigma = Sigma + int(y[j] ==  k+1)*transpose(temp)*temp/(N-K)
print "Sigma= ",Sigma
```

➢ Sigma= [[0.45377537 -0.20765221 -0.18936771 -0.00637846 -0.13880548 0.17363678 0.01796895 0.21276674 0.08443115 -0.09676817]
...
[-0.09676817 -0.04013298 0.14516513 0.07559054 0.00981541 -0.10780008 -0.07055438 -0.09583447 0.00462418 0.29821106]]

LDA in NumPy (3)

```
# discriminant function
invSigma = linalg.inv(Sigma)
deltak = zeros((N,K))
for j in range(N):
  for k in range(K):
    deltak[j,k] = (X[j:j+1,:]  -
   0.5*mu[k:k+1,:])*invSigma*transpose(mu[k:k+1,:]) + log(pi[k])
print "deltak= ",deltak
```

```
➢ deltak=  [[ 39.79481497  36.84476325  32.49859862 ...,
   31.37082131  29.29604055  35.22401622]
 …
 [ 28.17017203  33.08597263  31.60804467 ...,  34.65899072
   33.45568125 34.71614141]]
```


LDA: Error Rate

```
#classify
deltakT = transpose(deltak)
yhatT = matrix(deltakT.argmax(0) + 1)
yhat = transpose(yhatT)
errorrate = (0. + N - sum(y == yhat))/N
print "errorrate= ",errorrate
```

```
➢ errorrate=  0.316287878788
```

Program in 'vowel_lda.py'

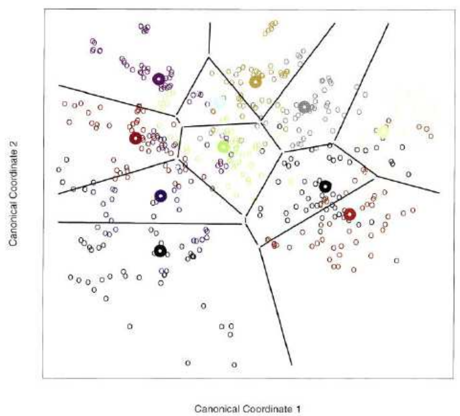

FIGURE 4.11. *Decision boundaries for the vowel training data, in the two-dimensional subspace spanned by the first two canonical variates. Note that in any higher-dimensional subspace, the decision boundaries are higher-dimensional affine planes, and could not be represented as lines.*

Exercise 8.2

Check the error rate of LDA (as fitted to the training data) on the vowel test data, and compare with the result of fitting the model to the test data.

Logistic Regression

- We will consider 2 classes only (the optimization gets hairy otherwise).
- Want to model the posterior probabilities of the 2 classes as linear functions in x, AND have them sum to 1 and be positive.
- The model is

$$\log \frac{\Pr(G=1 \mid X=x)}{\Pr(G=K \mid X=x)} = \beta_0 + \beta^T x$$

(remember the probabilities sum to 1)

- which is specified as a log-odds or logit transformation.

Logistic regression (ctd.)

A simple calculation shows

$$\Pr(G=1 \mid X=x) = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T x)}$$

$$\Pr(G=2 \mid X=x) = \frac{1}{1 + \exp(\beta_0 + \beta^T x)}$$

which clearly sum to 1.

We usually denote the probabilities as $\Pr(G=k|X=x) = p_k(x;\theta)$,

where θ is the set of all the β parameters.

Fitting logistic regression models

Fit by using maximum likelihood for the multinomial distribution

$$\log L = \sum_{i=1}^{N} \log p_{g_i}(x_i;\theta)$$

For the two-class case, code the responses as 0-1 (y = 1 when g = 1 and y=0 when g=2) then

$$\log L = \sum_{i=1}^{N} \{ y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta)) \}$$
$$= \sum_{i=1}^{N} \{ y_i \beta^T x_i - \log(1+e^{\beta^T x_i}) \}$$

Maximising the likelihood (2 classes)

Setting the derivatives to 0, we get the *score* equations

$$\frac{\partial \log L}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i,\beta)) = 0$$

(p+1 nonlinear equations in β). Solve via Newton-Raphson, using the Hessian

$$\frac{\partial^2 \log L}{\partial\beta\partial\beta^T} = -\sum_{i=1}^{N} x_i x_i^T p(x_i;\beta)[1-p(x_i;\beta)]$$

Starting with β^{old}, a single N-R update is

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 \log L}{\partial\beta\partial\beta^T} \right)^{-1} \frac{\partial \log L}{\partial \beta}$$

With derivatives evaluated at β^{old}. An initial value $\beta = 0$ should converge. If overshoots, use step-size halving.

SA Heart Disease Data

Class is presence or absence of coronary heart disease (chd), with predictors:

- ‘sbp’ - systolic blood pressure
- tobacco - cumulative tobacco (kg)
- ldl - low density lipoprotein cholesterol
- adiposity
- famhist - family history of heart disease (Present, Absent)
- typea type-A behavior
- obesity
- alcohol - current alcohol consumption
- age - age at onset

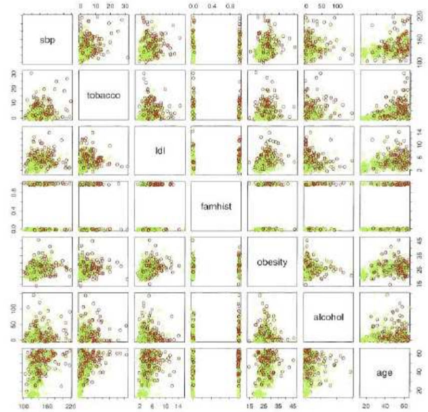

FIGURE 4.12. *A scatterplot matrix of the South African heart disease data. Each plot shows a pair of risk factors, and the cases and controls are color coded (red is a case). The variable* family history of heart disease (**famhist**) *is binary (yes or no).*

SA Heart Disease Data

```
data = loadtxt('saheartdis.txt')
P = 9
data = data.reshape(-1,P+1)

## reformat data as X and Y
y = data[:,9:10]
N = len(y)
K = 2
X = data[:,0:9]
X1 = concatenate((ones((shape(X)[0],1)),X),axis=1)
X1 = matrix(X1)
```


Logistic regression in python

```
# initialize
betaold = zeros((P+1,1))
tol = 0.000001
stepdiff = 1
while stepdiff > tol:
    p = exp(X1*betaold)
    p = p/(1+p)
    temp = multiply(p,(ones(shape(p)) - p))
    W = zeros((N,N))
    for j in range(N):
        W[j,j] = temp[j,0]
    beta = betaold +
  (linalg.inv(transpose(X1)*W*X1))*transpose(X1)*(matri
  x(y) - p)
    stepdiff = abs(betaold - beta).max()
    betaold = beta
```

Example

```
print 'beta= ',beta
➢      beta=  [[ -6.15072086e+00]
 [  6.50401713e-03]
 [  7.93764457e-02]
 [  1.73923898e-01]
 [  1.85865682e-02]
 [  9.25370419e-01]
 [  3.95950250e-02]
 [ -6.29098693e-02]
 [  1.21662401e-04]
 [  4.52253496e-02]]

p = exp(X1*beta)
p = p/(1+p)
print 'y,p(Y=1)= ',concatenate((y,p),axis = 1)

➢      y,p(Y=1)=  [[ 1.          0.71218288]
 [ 1.          0.33101091]
...
[ 0.          0.56521383]
 [ 1.          0.66884212]]
```


Classification

```
#classify
yhat = (p >= 0.5)
errorrate = (0. + N - sum(y == yhat))/N
print "errorrate= ",errorrate

➢errorrate=  0.266233766234
```

Program in 'heart_logreg.py'

Exercise 8.3

Using the South African Heart disease data, classify the elements using

a)Linear regression of the indicator matrix

b)LDA

and compare the results with those obtained from logistic regression.

Tests of Significance

$$\mathrm{cov}(\hat{\beta}) = ({}_1X^T W {}_1X)^{-1}$$

where W = diag($p_i(1-p_i)$). Thus the z-score for a predictor is

$$z_i = \frac{\beta_i}{\sqrt{v_i}}$$

where v_i is the ith diagonal element of cov(β).

In NumPy

```
# check z-scores
temp = multiply(p,(ones(shape(p)) - p))
W = zeros((N,N))
for j in range(N):
    W[j,j] = temp[j,0]
varbeta = linalg.inv(transpose(X1)*W*X1)
z =
  multiply(beta,1/transpose(matrix(sqrt(diag(varbeta))
  )))
print 'beta, z = ',concatenate((betahat0,z),axis=1)
```


Heart Data

```
beta, z =  [[ -6.15072086e+00  -4.70145121e+00]
 [  6.50401713e-03   1.13500271e+00] (blood pressure)
 [  7.93764457e-02   2.98375797e+00] (tobacco)
 [  1.73923898e-01   2.91516643e+00] (ldl)
 [  1.85865682e-02   6.34583236e-01] (adiposity)
 [  9.25370419e-01   4.06052972e+00] (famhist)
 [  3.95950250e-02   3.21382259e+00] (typea)
 [ -6.29098693e-02  -1.42176447e+00] (obesity)
 [  1.21662401e-04   2.71372912e-02] (alcohol)
 [  4.52253496e-02   3.72846429e+00]] (age)
```

A few surprises there, e.g., obesity is both insignificant, and negative!

Exercise 8.4

Perform logistic regression on each factor individually, and compare with the results on the previous slide.