Linear Methods for Classification

by

AINDRILA GARAI

MSC STATISTICS, IIT KANPUR

aindrilag22@iitk.ac.in

INTRODUCTION:

In this chapter, we study approaches for predicting qualitative responses, a process that is known as classification. Predicting classification a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category or class. We discuss some widely-used classifiers such as logistic regression, linear discriminant analysis, quadratic discriminant analysis, naive Bayes, and K-nearest neighbors.

Why Not Linear Regression:

There are at least two reasons not to perform classification using a regression method:

(a) A regression method cannot accommodate a qualitative response with more than two classes.
(b) It will not provide meaningful estimates of $Pr(Y|X)$, even with just two classes.

Logistic Regression:

Logistic regression models the probability that Y belongs to a particular category, then they may choose to use a lower threshold. Here we use the logistic function,

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

To fit the model, we use maximum likelihood method. This above quantity lies between 0 and 1. The logistic function will always produce an S-shaped curve (sigmoid curve).

After a bit of manipulation, we find that

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}$$

The quantity $p(X)/[1 - p(X)]$ is called the odds and can take on any value odds between 0 and ∞ indicating very low and very high probabilities respectively.

Next, by taking the logarithm of both sides of the above form, we arrive at

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

The left-hand side is called the log odds or logit which is linear in X. The rate of change in $p(X)$ per unit change in X depends on the current value of X.

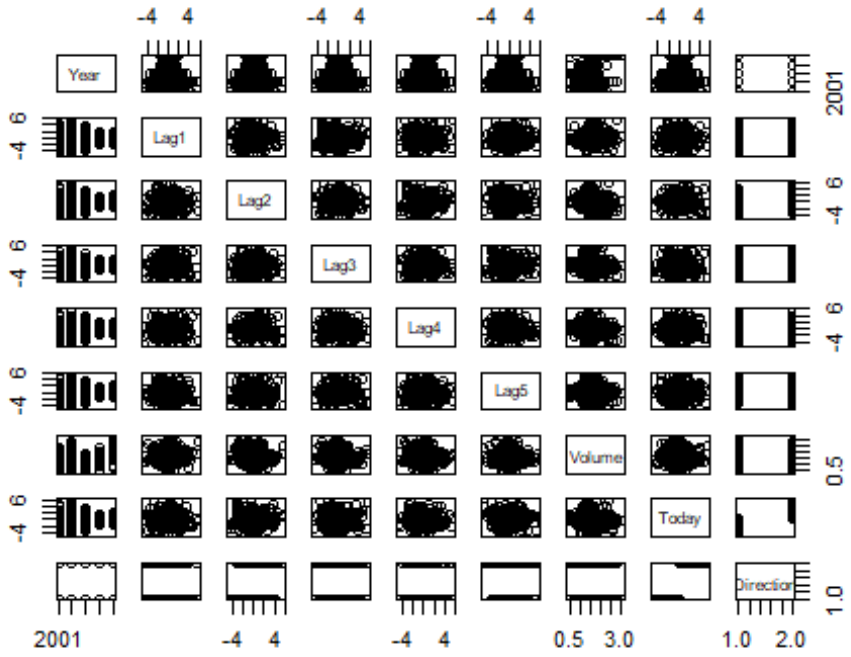
The estimates β_0 and β_1 are chosen to maximize this likelihood function.

In general, we take

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

where $X = (X_1, \ldots, X_p)$ are p predictors.

```
library (ISLR2)
data (Smarket)
pairs (Smarket,lwd=1) #  to plot Scatterplot matrix
```




```
cor(Smarket[, -9]) # removing the direction qualitative variable
```

```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
```

```
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

```
glm.fits <- glm (Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data
= Smarket, family = binomial) # in order to tell R to run a logistic regressi
on
summary (glm.fits)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

```
glm.probs <- predict (glm.fits , type = "response")
glm.probs [11:15]
```

```
##        11        12        13        14        15
## 0.4965211 0.5197834 0.5183031 0.4963852 0.4864892
```

```
contrasts (Smarket$Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
glm.pred <- rep ("Down", 1250)
glm.pred[glm.probs > .5] = "Up"
table (glm.pred ,Smarket$Direction)
```

```
##
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

```
mean (glm.pred == Smarket$Direction)
```

```
## [1] 0.5216
```

```
train <- (Smarket$Year < 2005) # true or false output
Smarket.2005 <- Smarket[!train , ] # give a submatrix
Direction.2005 <- Smarket$Direction[!train] # ! symbol can be used to reverse
all of the elements
```

```
glm.fits <- glm ( Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume , dat
a = Smarket , family = binomial , subset = train)
glm.probs <- predict (glm.fits , Smarket.2005, type = "response")
glm.fits
```

```
##
## Call:  glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket, subset = train)
##
## Coefficients:
## (Intercept)         Lag1         Lag2         Lag3         Lag4         La
g5
##    0.191213    -0.054178    -0.045805     0.007200     0.006441    -0.0042
23
##      Volume
##   -0.116257
##
## Degrees of Freedom: 997 Total (i.e. Null);  991 Residual
## Null Deviance:       1383
## Residual Deviance: 1381  AIC: 1395
```

```
glm.pred <- rep (" Down ", 252)
glm.pred[glm.probs > .5] <- "Up"
table (glm.pred , Direction.2005)
```

```
##         Direction.2005
## glm.pred Down Up
```

```
##     Down    77 97
##    Up       34 44
```

```
glm.fits <- glm (Direction ~ Lag1 + Lag2 , data = Smarket , family = binomial
, subset = train)
predict (glm.fits, newdata = data.frame (Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0
.8)), type = "response")
```

```
##         1         2
## 0.4791462 0.4960939
```

Multinomial Logistic Regression:

We sometimes wish to classify a response variable that has more than two classes. The extension of the two-class logistic regression approach into the setting of $K > 2$ classes is known as multinomial logistic regression. The mathematical form is -

$$\Pr(Y = k \mid X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+\cdots+\beta_{kp}x_p}}{1 + \sum_{l=1}^{K-1} e^{\beta_{l0}+\beta_{l1}x_1+\cdots+\beta_{lp}x_p}}$$

$$\text{for } k = 1, \ldots, K-1$$

Generative Models for Classification:

Reasons to use:

- When there is substantial separation between the two classes.
- Each of the classes and the sample size are small & more than two response classes.

Let π_k represents the prior probability that a randomly chosen observation comes from the prior kth class. Let $f_k(X) \equiv Pr(X|Y = k)$ denote the density function of X. Then Bayes' theorem states that-

$$\Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Instead of directly computing the posterior probability $p_k(x)$, we can simply plug in estimates of π_k and $f_k(X)$. In general, to estimate π_k we compute the fraction of the training observations that belong to the kth class.

Linear Discriminant Analysis:

In particular, we assume that $f_k(x)$ is normal or Gaussian.where μ_k and σ_k^2 are the mean and variance parameters for the kth class. Further assume that $\sigma_1^2 = \cdots = \sigma_k^2 = \sigma_k^2$.Then,

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}$$

where π_k denotes the prior probability that an observation belongs to the kth class,

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

Now, we assign the observation to the class for which it is largest.

If X is drawn from a Gaussian distribution within each class, to apply the Bayes classifier we still have to estimate the parameters $\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K$ and σ^2. The linear discriminant analysis (LDA) method approximates the Bayes classifier. In particular, the following estimates are used-

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

where n is the total number of training observations, and n_k is the number of training observations in the kth class.

In the case of p > 1 predictors, the LDA classifier assumes that the observations in the kth class are drawn from a multivariate Gaussian distribution $N(\mu_k, \sum)$, where μ_k is a class-specific mean vector, and $\sum$ is a covariance matrix that is common to all K classes.Again, Bayes classifier assigns an observation $X = x$ to the class for which is largest

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

```
library (MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```
lda.fit <- lda (Direction ~ Lag1 + Lag2 , data = Smarket , subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

```
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293

lda.pred <- predict (lda.fit , Smarket.2005)
lda.class <- lda.pred$class
sum (lda.pred$posterior[, 1] < .5)

## [1] 182
```

Remark:

- The LDA decision boundaries are pretty close to the Bayes decision boundaries impling that LDA is performing well on this data.
- The higher the ratio of parameters p to number of samples n, the more we expect this overfitting to play a role.
- **confusion matrix:**

A binary classifier can make two types of errors:

1. It can incorrectly assign an individual who defaults to the no default category
2. It can incorrectly assign an individual who does not default to the default category.

A confusion matrix is a convenient way to display this assigning to default category information.

- **low sensitivity:**

LDA is trying to approximate the Bayes classifier, which has the lowest total error rate out of all classifiers i.e. the Bayes classifier will yield the smallest possible total number of misclassified observations, regardless of the class from which the errors term. Varying the classifier threshold changes its true positive and false positive rate. These are also called the sensitivity.

- **ROC curve:** The ROC (receiver operating characteristic) curve is a popular graphic for simultaneously displaying two types of errors for all possible thresholds. The overall performance of a classifier summarized over all possible thresholds is given by the area under the (ROC) curve (AUC). An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.

Quadratic Discriminant Analysis:

The QDA classifier results from assuming that the observations from each class are drawn from a Gaussian distribution and plugging estimates for the parameters into Bayes' theorem in order to perform prediction. However, unlike LDA, QDA assumes that each class has its own covariance matrix. That is, it assumes that an observation from the kth class is of the form $X \sim N(\mu_k, \sum_k)$, where $\sum_k$ is a covariance matrix for the kth class.Under this

assumption, the Bayes classifier assigns an observation $X = x$ to the class for which is largest.

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\boldsymbol{\Sigma}_k^{-1}x + x^T\boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log\pi_k\end{aligned}$$

```
qda.fit <- qda (Direction ~ Lag1 + Lag2 , data = Smarket , subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

What to choose LDA OR QDA?

The answer lies in the bias-variance trade-of. LDA is a much less flexible classifier than QDA, and has lower variance. This can potentially lead to improved prediction performance. If LDA's assumption that the K classes share a common covariance matrix is off, then LDA can suffer from high bias. LDA tends to be a better if there are relatively few training observations.

Naive Bayes:

We assume here for $k = 1, \ldots, K$

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$$

where f_{kj} is the density function of the jth predictor among observations in the kth class.we completely eliminate the association between the p predictors. The naive Bayes assumption introduces some bias but reduces variance. Here,

$$\Pr(Y = k \mid X = x) = \frac{\pi_k \times f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)}{\sum_{l=1}^{K} \pi_l \times f_{l1}(x_1) \times f_{l2}(x_2) \times \cdots \times f_{lp}(x_p)}$$

for $k = 1, \ldots, K$

- If X_j is quantitative, then we can assume that $X_j|Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$. QDA with an additional assumption that the class-specific covariance matrix is diagonal.

- We can use a kernel density estimator, which is essentially a smoothed version of a histogram.
- If X_j is qualitative, then we can simply count the proportion of training observations for the jth predictor corresponding to each class.
- LDA is a special case of QDA with $c_{kjl} = 0$ for all $j = 1, \ldots, p$, $l = 1, \ldots, p$, and $k = 1, \ldots, K$. (Since LDA is simply a restricted version of QDA with $\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}$.)

```
library (e1071)
nb.fit <- naiveBayes (Direction ~ Lag1 + Lag2 , data = Smarket , subset = tra
in)
nb.fit
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     Down       Up
## 0.491984 0.508016
##
## Conditional probabilities:
##       Lag1
## Y             [,1]     [,2]
##   Down  0.04279022 1.227446
##   Up   -0.03954635 1.231668
##
##       Lag2
## Y             [,1]     [,2]
##   Down  0.03389409 1.239191
##   Up   -0.03132544 1.220765
```

```
nb.class <- predict (nb.fit , Smarket.2005)
table (nb.class , Direction.2005)
```

```
##         Direction.2005
## nb.class Down  Up
##     Down   28  20
##     Up     83 121
```