Logistic Regression

21/05

Recall that we are trying to solve a classification problem in which features x_i can be continuous or discrete (coded as 0/1) and the response y is discrete (0/1).

Logistic regression is a standard statistical method which can be used for classification.

It is analogous to linear regression for regression problems, but often performs better (in the sense that in a real-life problem, logistic regression is more likely to be a good method in practice.)

Recall [see last week's notes] that we can't just use the model

$$y_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

because this could make predictions for y_i outside (0,1). There are many ways of transforming a value in (-∞ , ∞) to a value in (0,1). [See board]. A common one in statistics is the *logistic transformation.*

Logistic Transformation

$$f(x) = \log\left(\frac{x}{1-x}\right)$$

Also called **log-odds.** If p is a probability of an event, p/(1-p) is called the **odds**. e.g. p=1/3 corresponds of odds of 1/2.

Note: the odds are called odds on in betting. Odds are usually quoted as odds against, which is p/(1-p). For example:

Ronnie O'Sullivan
2/1

Mark Selby
6/1

Neil Robertson
7/1

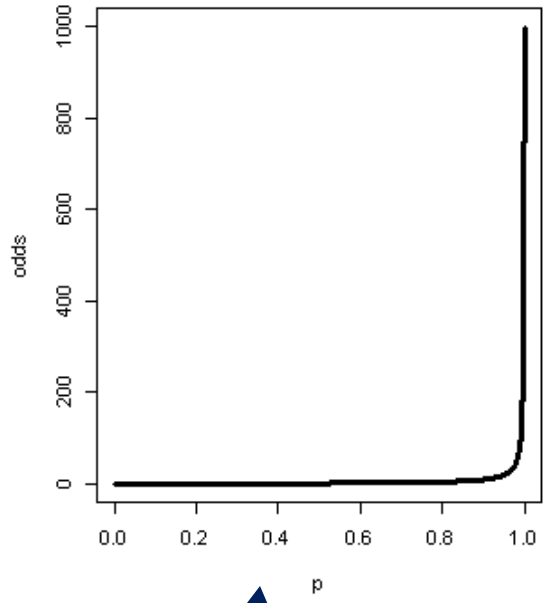
Ding Junhui
10/1

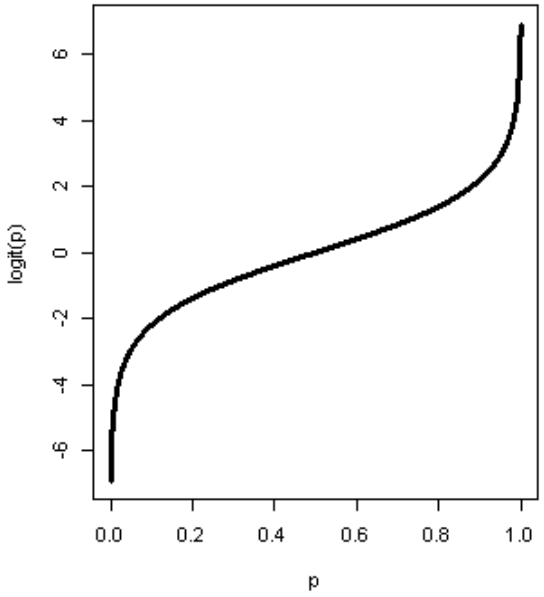
P(Ding Junhui wins) = p

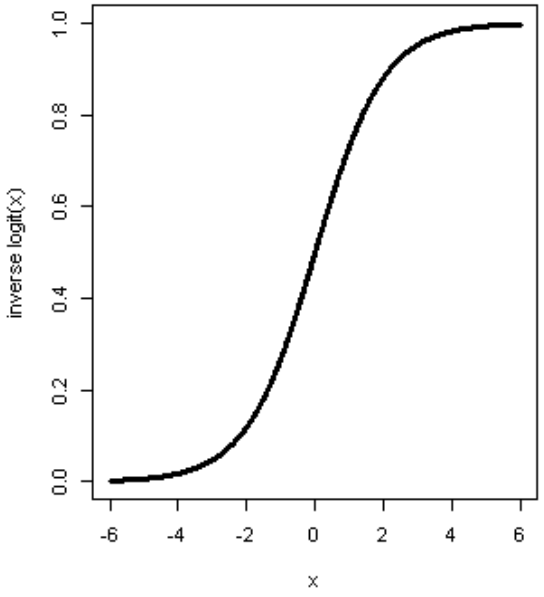
$$(1 - p)/p = 10$$

$$p = 1/11$$

Note: an event with a probability greater than ½ is quoted as “odds-on”. For example, a horse with a probability of winning of 2/3 has odds of 2, which is quoted as “2-1 on.” or “1-2”.







odds

logit(x) = f(x)

Inverse logit

- f is useful for transforming variables. Often if a variable is positive by nature, you should take its log. Similarly, if a variable takes values in (0,1), you should try taking its logit.
- Other possibilities: F(x) for any probability distribution f.
- Logit is particularly convenient because it means that various inferences from logistic regression can be expressed in closed form (similar to multiple linear regression requiring assumptions – which you should know!)

Logistic Regression

$$log\ \frac{P(y_i = 1)}{P(y_i = 0)} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Alternatively, could be written:

$$y_i \sim Bernoulli(logit^{-1}(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p))$$

Fitted in software by calculating the likelihood and maximising. (More difficult than linear regression.)

Interpreting β_i

The β_i are log(odds ratios). e^{β_i} is how much odds(y=1) increases with a one-unit increase in x_i (for x_i continuous).

To interpret this in terms of probability, you can roughly **divide by 4**. (Recommended in Gelman & Hill). This only works near *p=1/2,* by considering the slope of the logit curve [demonstrated in class.]

```
glm(formula = ifelse(sp == "B", 1, 0) ~ . - index, data = crabs)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.42869  -0.11730   0.01784   0.13141   0.34693

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.88303    0.09299   9.496  < 2e-16 ***
sexM        -0.13242    0.05027  -2.634 0.009116 **
FL          -0.26735    0.02692  -9.933  < 2e-16 ***
RW          -0.07513    0.02232  -3.365 0.000923 ***
CL          -0.04001    0.03545  -1.129 0.260457
CW           0.25063    0.02304  10.879  < 2e-16 ***
BD          -0.21638    0.03032  -7.137 1.88e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.0317067)

    Null deviance: 50.0000  on 199  degrees of freedom
Residual deviance:  6.1194  on 193  degrees of freedom
AIC: -113.8

Number of Fisher Scoring iterations: 2
```

Deviance = -2max value of LL

Measure of how well the model fits overall. Made up of the sum of the **deviance residuals**; used in place of ordinary (Pearson) residuals

```
Deviance Residuals:
     Min          1Q      Median          3Q         Max
-0.42869  -0.11730    0.01784    0.13141    0.34693
```

Confidence intervals for coefficients can be calculated using the normal distribution, eg.

For sexM:

```
-0.13242     0.05027
```

```
Null deviance: 50.0000  on 199  degrees of freedom
Residual deviance:  6.1194  on 193  degrees of freedom
AIC: -113.8
```

The deviance is supposed to go down by 1 for each variable added, if the variables are random noise. Under this assumption, the deviance will be chi-squared distributed with (199-193) degrees of freedom.

```
1-pchisq(50-6.11, df=199-193)
[1] 7.772806e-08
```

Analogue of ANOVA F-test for overall significance of the model.

Note: just like in linear regression, you should usually centre the variables before doing the analysis. This makes the intercept more meaningful and improves convergence of the algorithm which fits the model.

Note 2: As usual, SAS outputs quite a bit more. The extra table of statistics are measures of correlation between the fitted probabilities and the observed values of y. (Percent Concordant etc.)

Using logistic regression for classification

Recall that this section is about classifying things. We can get a 0/1 prediction from logistic regression just by rounding (up from 0.5, down from 0.5).

Sometimes you have a rare y=1 and you want to include 50% 1's and 50% 0's in your data for logistic regression. Then you should adjust the cutoff for prediction. This is called ***case-control sampling****.*

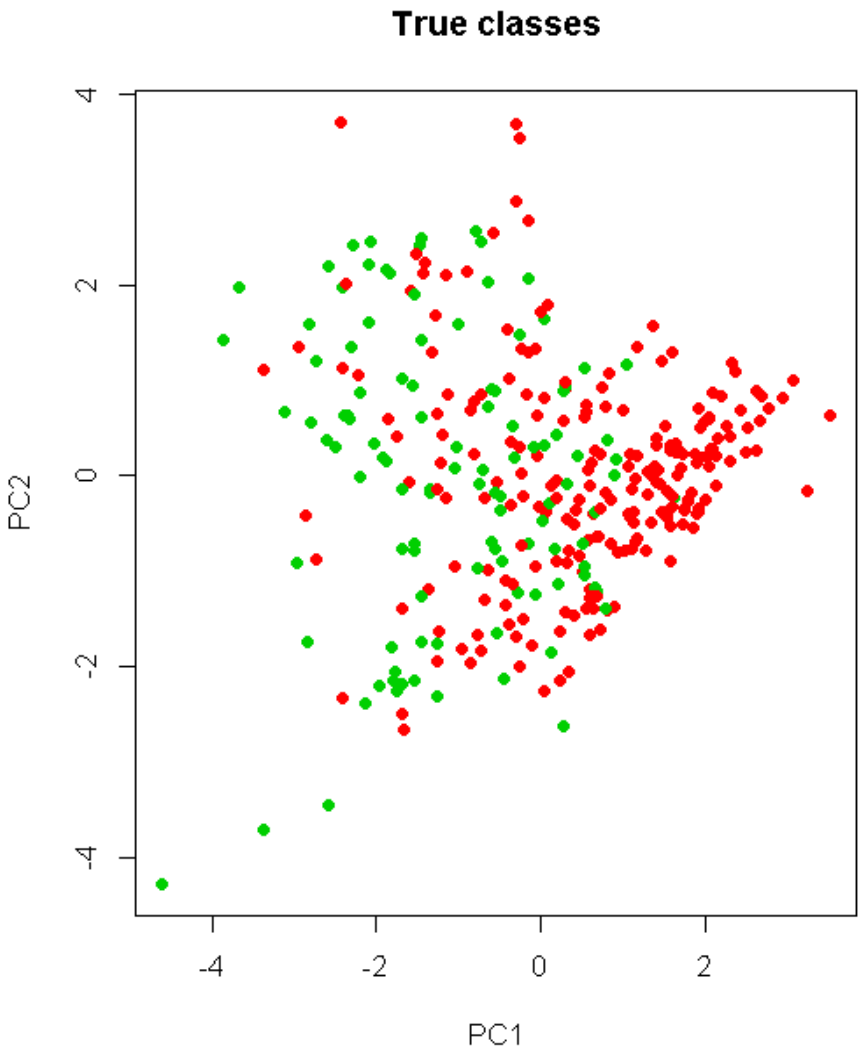
```
folds <- sample(1:5, 200, replace=T)
cv <- rep(0,5)

for (i in 1:5){
  train <- crabs[folds!=i,]
  test <- crabs[folds==i,]
  y <- ifelse(train$sp=="B", 1, 0)
  model <- glm(y ~., data=train[,-1], family="binomial")
  pred <- predict(model, test[,-1])
  pred <- 1/(1+exp(-pred))
  cv[i] <- sum(ifelse(pred>0.5,1,0) == ifelse(test[,1]=="B",1,0))/nrow(test)
}
```

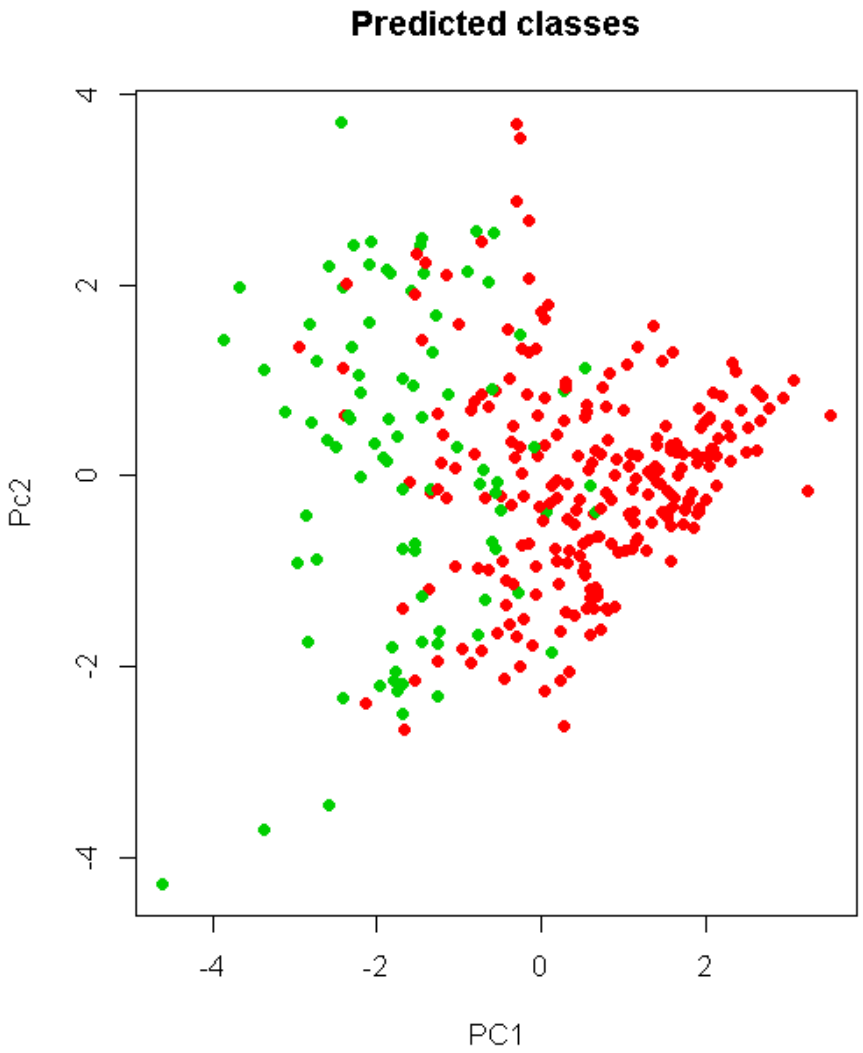
In the crabs data, the model works perfectly but logistic regression gives a warning. This is because the algorithm fails to converge if the classes are separated by a hyperplane (which is the case in this example.)

```
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Example (if time):

```
library(MASS)
data(Pima.tr)
data(Pima.te)
model <- glm(type ~., data=Pima.tr, family="binomial")
summary(model)
pred <- predict(model, Pima.te)
pred <- 1/(1+exp(-pred))
pred <- round(pred)
sum(pred==ifelse(Pima.te$type=="Yes",1,0))/nrow(Pima.te)
```


True classes
PC2
PC1
4
2
0
-2
-4
-4
-2
0
2


Predicted classes
Pc2
PC1
4
2
0
-2
-4
-4
-2
0
2