BUS41100 Applied Regression Analysis

Week 6: Binary Outcomes

Logistic Regression & Classification

Max H. Farrell
The University of Chicago Booth School of Business

Discrete Responses

So far, the outcome Y has been continuous, but many times we are interested in discrete responses:

- Binary: $Y = 0$ or 1
  - Buy or don't buy
- More categories: $Y = 0, 1, 2, 3, 4$
  - Unordered: buy product A, B, C, D, or nothing
  - Ordered: rate 1–5 stars
- Count: $Y = 0, 1, 2, 3, 4, \ldots$
  - How many products bought in a month?

Today we're only talking about binary outcomes

- By far the most common application
- Illustrate all the ideas
- Week 9 covers the rest

Binary response data

The goal is generally to predict the **probability that** $Y = 1$.
You can then do classification based on this estimate.

- Buy or not buy
- Win or lose
- Sick or healthy
- Pay or default
- Thumbs up or down

Relationship type questions are interesting too

- Does an ad increase $\mathbb{P}[\text{buy}]$?
- What type of patient is more likely to live?

Generalized Linear Model

What's wrong with our MLR model?

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

$Y = \{0, 1\}$ causes two problems:

1. Normal can be any number, how can $Y = \{0, 1\}$ only?
2. Can the conditional mean be linear?

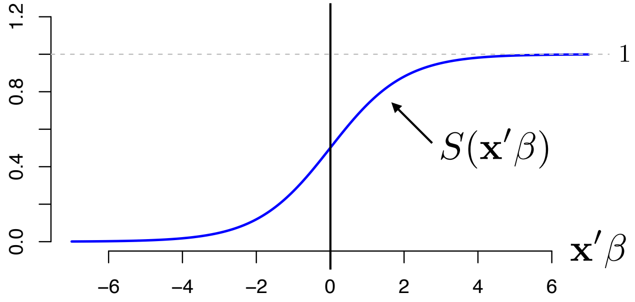
$$\begin{aligned}\mathbb{E}[Y|\boldsymbol{X}] &= \mathbb{P}(Y = 1|\boldsymbol{X}) \times 1 + \mathbb{P}(Y = 0|\boldsymbol{X}) \times 0 \\ &= \mathbb{P}(Y = 1|\boldsymbol{X})\end{aligned}$$

- We need a model that gives mean/probability values between 0 and 1.
- We'll use a transform function that takes the usual linear model and gives back a value between zero and one.

The generalized linear model is

$$\mathbb{P}(Y = 1 | X_1, \ldots, X_d) = S(\beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d)$$

where S is a *link* function that increases from zero to one.



$$S^{-1}\Big(\mathbb{P}(Y = 1 | X_1, \ldots, X_d)\Big) = \underbrace{\beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d}_{\text{Linear!}}$$

There are two main functions that are used for this:

- Logistic Regression: $S(z) = \dfrac{e^z}{1+e^z}$.
- Probit Regression: $S(z) = \texttt{pnorm}(z) = \Phi(z)$.

Both are S-shaped and take values in $(0,1)$.

Logit is usually preferred, but

they result in practically the same fit.

(These are only for binary outcomes, in week 9 we will see that other types of Y need different *link* functions $S(\cdot)$.)

Binary Choice Motivation

GLMs are motivated from a prediction/data point of view.
What about economics?

Standard **binary choice** model for an economic agent

- e.g. purchasing, market entry, repair/replace, ...

1. Take action if payoff is big enough: $Y = \mathbb{1}\{\text{utility} > \text{cost}\}$
2. Utility is linear $= Y^* = \beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d + \varepsilon$
3. $\varepsilon \sim$???
   - Probit GLM $\Leftrightarrow \varepsilon \sim \mathcal{N}(0, 1)$
   - Logit GLM $\Leftrightarrow \varepsilon \sim$ Logistic a.k.a. Type 1 Extreme value

(see `week6-Rcode.R`)

(We're skipping over lots of details, including behaviors, dynamics, etc.)

Logistic regression

We'll use logistic regression, such that

$$\mathbb{P}(Y = 1|X_1 \ldots X_d) = S\left(\mathbf{X}'\boldsymbol{\beta}\right) = \frac{\exp[\beta_0 + \beta_1 X_1 \ldots + \beta_d X_d]}{1 + \exp[\beta_0 + \beta_1 X_1 \ldots + \beta_d X_d]}.$$

These models are easy to fit in R:

```
glm(Y ~ X1 + X2, family=binomial)
```

- "g" is for generalized; `binomial` indicates $Y = 0$ or 1.
- Otherwise, `glm` uses the same syntax as `lm`.
- The "logit" link is more common, and is the default in R.

Interpretation

Model the probability:

$$\mathbb{P}(Y = 1 | X_1 \ldots X_d) = S\left(\mathbf{X}'\boldsymbol{\beta}\right) = \frac{\exp[\beta_0 + \beta_1 X_1 \ldots + \beta_d X_d]}{1 + \exp[\beta_0 + \beta_1 X_1 \ldots + \beta_d X_d]}.$$

Invert to get linear log odds ratio:

$$\log\left(\frac{\mathbb{P}(Y = 1 | X_1 \ldots X_d)}{\mathbb{P}(Y = 0 | X_1 \ldots X_d)}\right) = \beta_0 + \beta_1 X_1 \ldots + \beta_d X_d.$$

Therefore:

$$e^{\beta_j} = \frac{\mathbb{P}(Y = 1 | X_j = (x+1))}{\mathbb{P}(Y = 0 | X_j = (x+1))} \Bigg/ \frac{\mathbb{P}(Y = 1 | X_j = x)}{\mathbb{P}(Y = 0 | X_j = x)}$$

Repeating the formula:

$$e^{\beta_j} = \frac{\mathbb{P}(Y=1|X_j=(x+1))}{\mathbb{P}(Y=0|X_j=(x+1))} \Big/ \frac{\mathbb{P}(Y=1|X_j=x)}{\mathbb{P}(Y=0|X_j=x)}$$

Therefore:

- e^{β_j} = change in the odds for a one unit increase in X_j.
- …holding everything else constant, as always!
- Always $e^{\beta_j} > 0$, $e^0 = 1$. Why?

Odds Ratios & 2×2 Tables

Odds Ratios are easier to understand when X is also binary.
We can make a table and compute everything.

Example: Data from an online recruiting service

- Customers are **firms** looking to hire
- Fixed price is charged for access
  - Post job openings, find candidates, etc
- $X =$ `price` – price they were shown, \$99 or \$249
- $Y =$ `buy` – did this firm sign up for service: yes/no

```
> price.data <- read.csv("priceExperiment.csv")
> table(price.data$buy, price.data$price)

     99  249
  0  912 1026
  1  293  132
```

With the 2×2 table, we can compute everything!

- probabilities: $\mathbb{P}[Y = 1 \mid X = 99] = \dfrac{293}{293 + 912}$

  $\Rightarrow$ 25% of people buy at \$99

- odds ratios: $\dfrac{\mathbb{P}[Y = 1 \mid X = 99]}{\mathbb{P}[Y = 0 \mid X = 99]} = \dfrac{\frac{293}{293+912}}{\frac{912}{293+912}} = \dfrac{293}{912}$

  $\Rightarrow$ don't buy is $75\%/25\% = 3\times$ more likely vs buy at \$99

- even coefficients!

$$e^{(249-99)b_1} = \frac{\mathbb{P}(Y = 1|X = 249)}{\mathbb{P}(Y = 0|X = 249)} \Bigg/ \frac{\mathbb{P}(Y = 1|X = 99)}{\mathbb{P}(Y = 0|X = 99)}$$
$$= 0.40$$

$\Rightarrow$ Price $\uparrow$ \$150 $\rightarrow$ odds of buying 40% of what they were

$\Rightarrow$ Price $\downarrow$ \$150 $\rightarrow$ odds of buying $1/0.4 = 2.5\times$ higher

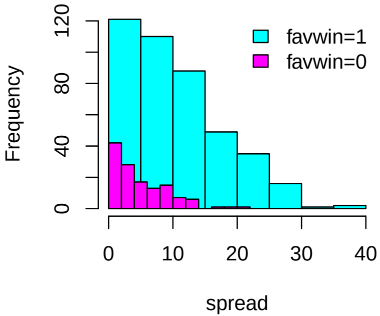
Logistic regression

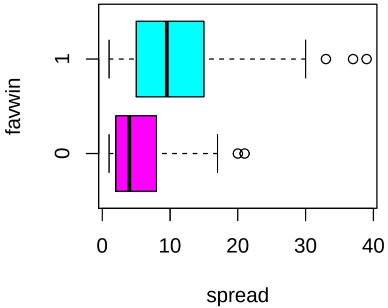
Continuous X means no more tables

- Same interpretation, different visualization

Example: Las Vegas betting point spreads for 553 NBA games and the resulting scores.

- Response: `favwin=1` if favored team wins.
- Covariate: `spread` is the Vegas point spread.





This is a weird situation where we assume no intercept.

- Most likely the Vegas betting odds are efficient.
- A spread of zero implies $p(\mathrm{win}) = 0.5$ for each team.

We get this out of our model when $\beta_0 = 0$

$$\mathbb{P}(\mathrm{win}) = \exp[\beta_0]/(1 + \exp[\beta_0]) = 1/2.$$

The model we want to fit is thus

$$\mathbb{P}(\mathrm{favwin}|\mathrm{spread}) = \frac{\exp[\beta_1 \times \mathrm{spread}]}{1 + \exp[\beta_1 \times \mathrm{spread}]}.$$

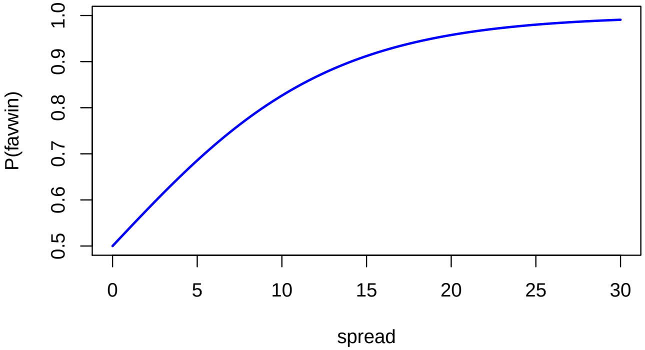
R output from `glm`:

```
> nbareg <- glm(favwin~spread-1, family=binomial)
> summary(nbareg)  ## abbreviated output

Coefficients:
       Estimate Std. Error z value Pr(>|z|)
spread  0.15600    0.01377   11.33   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

    Null deviance: 766.62  on 553  degrees of freedom
Residual deviance: 527.97  on 552  degrees of freedom
AIC: 529.97
```

Interpretation

The fitted model is

$$\hat{\mathbb{P}}(\text{favwin}|\text{spread}) = \frac{\exp[0.156 \times \text{spread}]}{1 + \exp[0.156 \times \text{spread}]}.$$



Convert to odds-ratio

```
> exp(coef(nbareg))
  spread
1.168821
```

- A 1 point increase in the spread means the favorite is 1.17 times more likely to win
- What about a 10-point increase: `exp(10*coef(nbareg))` ≈ 4.75 times more likely

Uncertainty:

```
> exp(confint(nbareg))
Waiting for profiling to be done...
   2.5 %   97.5 %
1.139107 1.202371
```

Code: `exp(cbind(coef(logit.reg), confint(logit.reg)))`

New predictions

The `predict` function works as before, but add `type = "response"` to get $\hat{\mathbb{P}} = \exp[\mathbf{x}'\mathbf{b}]/(1 + \exp[\mathbf{x}'\mathbf{b}])$ (otherwise it just returns the linear function $\mathbf{x}'\mathbf{b}$).

Example: Chicago vs Sacramento spread is SK by 1

$$\hat{\mathbb{P}}(\text{CHI win}) = \frac{1}{1 + \exp[0.156 \times 1]} = 0.47$$

- Orlando (-7.5) at Washington: $\hat{\mathbb{P}}(favwin) = 0.76$
- Memphis at Cleveland (-1): $\hat{\mathbb{P}}(favwin) = 0.53$
- Golden State at Minnesota (-2.5): $\hat{\mathbb{P}}(favwin) = 0.60$
- Miami at Dallas (-2.5): $\hat{\mathbb{P}}(favwin) = 0.60$

Investigate our efficiency assumption: we know the favorite usually wins but do they cover the spread?

```
> cover <- (favscr > (undscr + spread))
> table(cover)

FALSE  TRUE
  280   273
```

About 50/50, as expected, but is it predictable?

```
> summary(glm(cover ~ spread, family=binomial))$coefficients
                Estimate Std. Error     z value  Pr(>|z|)
(Intercept)  0.004479737 0.14059905  0.03186179 0.9745823
spread      -0.003100138 0.01164922 -0.26612406 0.7901437
```

Classification

A common goal with logistic regression is to classify the inputs depending on their predicted response probabilities.

Example: evaluating the credit quality of (potential) debtors.

- Take a list of borrower characteristics.
- Build a prediction rule for their credit.
- Use this rule to automatically evaluate applicants (and track your risk profile).

You can do all this with logistic regression, and then use the predicted probabilities to build a classification rule.

- A simple classification rule would be that anyone with $\hat{\mathbb{P}}(\text{good}|\mathbf{x}) > 0.5$ can get a loan, and the rest cannot.

(Classification is a huge field, we're only scratching the surface here.)

We have data on 1000 loan applicants at German community banks, and judgment of the loan outcomes (good or bad).

The data has 20 borrower characteristics, including

- credit history (5 categories),
- housing (rent, own, or free),
- the loan purpose and duration,
- and installment rate as a percent of income.

Unfortunately, many of the columns in the data file are coded categorically in a very opaque way. (Most are `factors` in R.)

Logistic regression yields $\hat{\mathbb{P}}[\text{good}|\mathbf{x}] = \hat{\mathbb{P}}[Y = 1|\mathbf{x}]$:

```
> full <- glm(GoodCredit~., family=binomial, data=credit)
> predfull <- predict(full, type="response")
```

Need to compare to binary $Y = \{0, 1\}$.

- Convert: $\hat{Y} = \mathbb{1}\{\hat{\mathbb{P}}[Y = 1|\mathbf{x}] > 0.5\}$
- classification error: $Y_i - \hat{Y}_i = \{-1, 0, 1\}$.

```
> errorfull <- credit[,1] - (predfull >= .5)
> table(errorfull)
 -1   0   1
 74 786 140
> mean(abs(errorfull))        ## add weights if you want
[1] 0.214
> mean(errorfull^2)
[1] 0.214
```

We'll compare a couple different models. Next week we'll build more models.

```
> empty <- glm(GoodCredit~1, family=binomial, data=credit)
> history <- glm(GoodCredit~history3, family=binomial, data=cred
> full <- glm(GoodCredit~., family=binomial, data=credit)
```

We want to compare the accuracy of their predictions. But how do we compare binary $Y = \{0, 1\}$ to a probability?

- We compare misclassification rates:

```
> c(full=mean(abs(errorfull)),
+     history=mean(abs(errorhistory)),
+     empty=mean(abs(errorempty)) )
   full history   empty
  0.214   0.283   0.300
```

Why is this both obvious and not helpful?

A word of caution

Why not just throw everything in there?

```
> too.good <- glm(GoodCredit~. + .^2, family=binomial,
+ data=credit)
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

This warning means you have the logistic version of our "connect the dots" model.

- Just as useless as before!

```
> c(empty=mean(abs(errorempty)),
+   history=mean(abs(errorhistory)),
+   full=mean(abs(errorfull)) ,
+   too.good=mean(abs(errortoo.good)) )
   empty  history     full too.good
   0.300    0.283    0.214    0.000
```

ROC & PR curves

You can also do classification with cut-offs other than $1/2$.

- Suppose the risk associated with one action is higher than for the other.
- You'll want to have $p > 0.5$ of a positive outcome before taking the risky action.

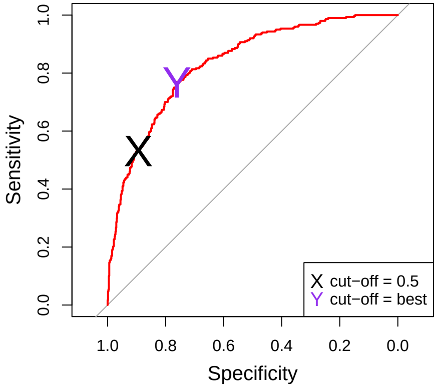
We want to know:

- What happens as the cut-off changes?
- Is there a "best" cut-off?

One way is to answer is by looking at two curves:

1. ROC: Receiver Operating Characteristic
2. PR: Precision-Recall

```
> library("pROC")
> roc.full <- roc(credit[,1] ~ predfull)
> coords(roc.full, x=0.5)
  threshold specificity sensitivity
  0.5000000   0.8942857   0.5333333
> coords(roc.full, "best")
  threshold specificity sensitivity
  0.3102978   0.7614286   0.7700000
```



Sensitivity

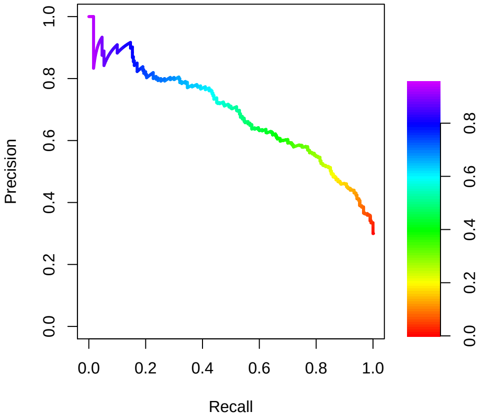
true positive rate

Specificity

true negative rate

Many related names: hit rate, fall-out false discovery rate, . . .

```
> library("PRROC")
> pr.full <- pr.curve(scores.class0=predfull,
+         weights.class0=credit[,1], curve=TRUE)
```



Recall

true positive rate

same as senstivity

Precision

positive predictive value

Many related names: hit rate, fall-out false discovery rate, . . .

Summary

We changed Y from continuous to binary.

- As a result we had to change everything
  - model, interpretation, . . .
- But still linear regression
  - Same goals: predictions, relationships
  - Same concerns: visualization, overfitting

In week 9 we will extend what we learned today to:

- Other discrete outcomes, using generalized linear models

Coming Up Next

Next week:

- Proposal
- Model Building

Week 8:

- Time series data

Weeks 9:

- More on discrete outcomes

Week 10:

- FINAL
- Projects Due