Regression 3: Logistic Regression

Marco Baroni

Practical Statistics in R

Outline

Outline

Outline

Modeling discrete response variables

- In a very large number of problems in cognitive science and related fields
  - the response variable is categorical, often *binary* (yes/no; acceptable/not acceptable; phenomenon takes place/does not take place)
  - potentially explanatory factors (independent variables) are categorical, numerical or both

Examples: binomial responses

- Is linguistic construction X rated as "acceptable" in the following condition(s)?
- Does sentence S, that has features Y, W and Z, display phenomenon X? (linguistic corpus data!)
- Is it common for subjects to decide to purchase the good X given these conditions?
- Did subject make more errors in this condition?
- How many people answer YES to question X in the survey
- Do old women like X more than young men?
- Did the subject feel pain in this condition?
- How often was reaction X triggered by these conditions?
- Do children with characteristics X, Y and Z tend to have autism?

Examples: multinomial responses

- Discrete response variable with natural ordering of the levels:
  - Ratings on a 6-point scale
    - Depending on the number of points on the scale, you might also get away with a standard linear regression
  - Subjects answer YES, MAYBE, NO
  - Subject reaction is coded as FRIENDLY, NEUTRAL, ANGRY
  - The cochlear data: experiment is set up so that possible errors are *de facto* on a 7-point scale
- Discrete response variable without natural ordering:
  - Subject decides to buy one of 4 different products
  - We have brain scans of subjects seeing 5 different objects, and we want to predict seen object from features of the scan
  - We model the chances of developing 4 different (and mutually exclusive) psychological syndromes in terms of a number of behavioural indicators

Binomial and multinomial logistic regression models

- Problems with binary (yes/no, success/failure, happens/does not happen) dependent variables are handled by (binomial) logistic regression
- Problems with more than one discrete output are handled by
  - ordinal logistic regression, if outputs have natural ordering
  - multinomial logistic regression otherwise
- The output of ordinal and especially multinomial logistic regression tends to be hard to interpret, whenever possible I try to reduce the problem to a binary choice
  - E.g., if output is yes/maybe/no, treat "maybe" as "yes" and/or as "no"
- Here, I focus entirely on the binomial case

Don't be afraid of logistic regression!

- Logistic regression seems less popular than linear regression
- This might be due in part to historical reasons
  - the formal theory of generalized linear models is relatively recent: it was developed in the early nineteen-seventies
  - the iterative maximum likelihood methods used for fitting logistic regression models require more computational power than solving the least squares equations
- Results of logistic regression are not as straightforward to understand and interpret as linear regression results
- Finally, there might also be a bit of prejudice against discrete data as less "scientifically credible" than hard-science-like continuous measurements

Don't be afraid of logistic regression!

- Still, if it is natural to cast your problem in terms of a discrete variable, you should go ahead and use logistic regression
- Logistic regression might be trickier to work with than linear regression, but it's still much better than pretending that the variable is continuous or artificially re-casting the problem in terms of a continuous response

The Machine Learning angle

- *Classification* of a set of observations into 2 or more discrete categories is a central task in Machine Learning
- The classic *supervised learning* setting:
  - Data points are represented by a set of *features*, i.e., discrete or continuous explanatory variables
  - The "training" data also have a *label* indicating the class of the data-point, i.e., a discrete binomial or multinomial dependent variable
  - A model (e.g., in the form of weights assigned to the dependent variables) is fitted on the training data
  - The trained model is then used to predict the class of unseen data-points (where we know the values of the features, but we do not have the label)

The Machine Learning angle

- Same setting of logistic regression, except that emphasis is placed on predicting the class of unseen data, rather than on the significance of the effect of the features/independent variables (that are often too many – hundreds or thousands – to be analyzed singularly) in discriminating the classes
- Indeed, logistic regression is also a standard technique in Machine Learning, where it is sometimes known as Maximum Entropy

Outline

- Logistic regression
  - Introduction
  - The model
  - Looking at and comparing fitted models
- Logistic regression in R

Classic multiple regression

- The by now familiar model:

$$y = \beta_0 + \beta_1 \times \mathrm{x}_1 + \beta_2 \times \mathrm{x}_2 + ... + \beta_n \times \mathrm{x}_\mathrm{n} + \epsilon$$

- Why will this not work if variable is binary (0/1)?
- Why will it not work if we try to model proportions instead of responses (e.g., proportion of YES-responses in condition C)?

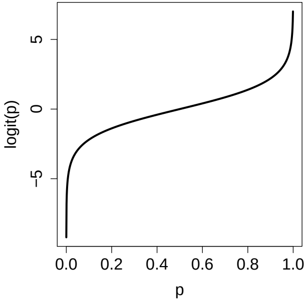
Modeling log odds ratios

- Following up on the “proportion of YES-responses” idea, let’s say that we want to model the *probability* of one of the two responses (which can be seen as the population proportion of the relevant response for a certain choice of the values of the dependent variables)
- Probability will range from 0 to 1, but we can look at the *logarithm of the odds ratio instead*:

$$\mathit{logit}(p) = \log \frac{p}{1-p}$$

- This is the logarithm of the ratio of probability of 1-response to probability of 0-response
  - It is arbitrary what counts as a 1-response and what counts as a 0-response, although this might hinge on the ease of interpretation of the model (e.g., treating YES as the 1-response will probably lead to more intuitive results than treating NO as the 1-response)
- Log odds ratios are not the most intuitive measure (at least for me), but they range continuously from $-\infty$ to $+\infty$

From probabilities to log odds ratios



The logistic regression model

- Predicting log odds ratios:

$$logit(p) = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n$$

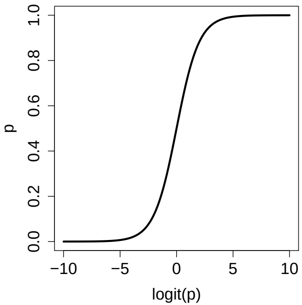
- Back to probabilities:

$$p = \frac{e^{logit(p)}}{1 + e^{logit(p)}}$$

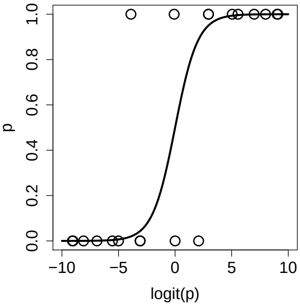
- Thus:

$$p = \frac{e^{\beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n}}{1 + e^{\beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n}}$$

From log odds ratios to probabilities



Probabilities and responses



A subtle point: no error term

- NB:

$$logit(p) = \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 + ... + \beta_n \times x_n$$

- The outcome here is not the observation, but (a function of) p, the expected value of the *probability* of the observation given the current values of the dependent variables
- This probability has the classic “coin tossing” Bernoulli distribution, and thus variance is not free parameter to be estimated from the data, but model-determined quantity given by $p(1 - p)$
- Notice that errors, computed as $\text{observation} - p$, are not independently normally distributed: they must be near 0 or near 1 for high and low ps and near .5 for ps in the middle

The generalized linear model

- Logistic regression is an instance of a "generalized linear model"
- Somewhat brutally, in a generalized linear model
  - a weighted linear combination of the explanatory variables models a function of the expected value of the dependent variable (the "link" function)
  - the actual data points are modeled in terms of a distribution function that has the expected value as a parameter
- General framework that uses same fitting techniques to estimate models for different kinds of data

Linear regression as a generalized linear model

- Linear prediction of a function of the mean:

$$g(E(y)) = X\beta$$

- "Link" function is identity:

$$g(E(y)) = E(y)$$

- Given mean, observations are normally distributed with variance estimated from the data
  - This corresponds to the error term with mean 0 in the linear regression model

Logistic regression as a generalized linear model

- Linear prediction of a function of the mean:

$$g(E(y)) = X\beta$$

- "Link" function is :

$$g(E(y)) = \log \frac{E(y)}{1 - E(y)}$$

- Given $E(y)$, i.e., p, observations have a Bernoulli distribution with variance $p(1 - p)$

Estimation of logistic regression models

- Minimizing the sum of squared errors is not a good way to fit a logistic regression model
- The least squares method is based on the assumption that errors are normally distributed and independent of the expected (fitted) values
- As we just discussed, in logistic regression errors depend on the expected (p) values (large variance near .5, variance approaching 0 as p approaches 1 or 0), and for each p they can take only two values ($1 - p$ if response was 1, $p - 0$ otherwise)

Estimation of logistic regression models

- The β terms are estimated instead by maximum likelihood, i.e., by searching for that set of βs that will make the observed responses maximally likely (i.e., a set of β that will in general assign a high *p* to 1-responses and a low *p* to 0-responses)
- There is no closed-form solution to this problem, and the optimal $\vec{\beta}$ tuning is found with iterative "trial and error" techniques
  - Least-squares fitting is finding the maximum likelihood estimate for linear regression and *vice versa* maximum likelihood fitting is done by a form of *weighted* least squares fitting

Outline

Interpreting the βs

- Again, as a rough-and-ready criterion, if a β is more than 2 standard errors away from 0, we can say that the corresponding explanatory variable has an effect that is significantly different from 0 (at $\alpha = 0.05$)
- However, *p* is not a linear function of $X\beta$, and the same β will correspond to a more drastic impact on *p* towards the center of the *p* range than near the extremes (recall the S shape of the *p* curve)
- As a rule of thumb (the "divide by 4" rule), $\beta/4$ is an upper bound on the difference in *p* brought about by a unit difference on the corresponding explanatory variable

Goodness of fit

- Again, measures such as R^2 based on residual errors are not very informative
- One intuitive measure of fit is the *error rate*, given by the proportion of data points in which the model assigns $p > .5$ to 0-responses or $p < .5$ to 1-responses
  - This can be compared to baseline in which the model always predicts 1 if majority of data-points are 1 or 0 if majority of data-points are 0 (baseline error rate given by proportion of minority responses over total)
- Some information lost (a .9 and a .6 prediction are treated equally)
- Other measures of fit proposed in the literature, no widely agreed upon standard

Binned goodness of fit

- Goodness of fit can be inspected visually by grouping the *p*s into equally wide bins (0-0.1,0.1-0.2, ...) and plotting the average *p* predicted by the model for the points in each bin vs. the observed proportion of 1-responses for the data points in the bin
- We can also compute a R^2 or other goodness of fit measure on these binned data

Deviance

- Deviance is an important measure of fit of a model, used also to compare models
- Simplifying somewhat, the deviance of a model is -2 times the log likelihood of the data under the model
  - plus a constant that would be the same for all models for the same data, and so can be ignored since we always look at differences in deviance
- The larger the deviance, the worse the fit
- As we add parameters, deviance decreases

Deviance

- The difference in deviance between a simpler and a more complex model approximates a χ^2 distribution with the difference in number of parameters as df's
  - This leads to the handy rule of thumb that the improvement is significant (at $\alpha = .05$) if the deviance difference is larger than the parameter difference (play around with `pchisq()` in R to see that this is the case)
- A model can also be compared against the "null" model that always predicts the same *p* (given by the proportion of 1-responses in the data) and has only one parameter (the fixed predicted value)

Outline

Outline

Logistic regression

Logistic regression in R

Preparing the data and fitting the model

Practice

Back to the Graffeo et al.'s discount study

Fields in the `discount.txt` file

subj	Unique subject code
sex	M or F
age	NB: contains some NA
presentation	*absdiff* (amount of discount), *result* (price after discount), *percent* (percentage discount)
product	*pillow*, (camping) *table*, *helmet*, (bed) *net*
choice	Y (buys), N (does not buy) → the discrete response variable

Preparing the data

- Read the file into an R data-frame, look at the summaries, etc.
- Note in the summary of `age` that R "understands" NAs (i.e., it is not treating age as a categorical variable)
- We can filter out the rows containing NAs as follows:

```
> e<-na.omit(d)
```

- Compare summaries of `d` and `e`
  - `na.omit` can also be passed as an option to the modeling functions, but I feel uneasy about that
- Attach the NA-free data-frame

Logistic regression in R

```
> sex_age_pres_prod.glm<-glm(choice~sex+age+
  presentation+product,family="binomial")

> summary(sex_age_pres_prod.glm)
```

Selected lines from the `summary()` output

- Estimated β coefficients, standard errors and z scores (β/std. error):

```
Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
sexM                 -0.332060   0.140008  -2.372  0.01771
age                  -0.012872   0.006003  -2.144  0.03201
presentationpercent   1.230082   0.162560   7.567 3.82e-14
presentationresult    1.516053   0.172746   8.776  < 2e-16
```

- Note automated creation of binary dummy variables: discounts presented as percents and as resulting values are significantly more likely to lead to a purchase than discounts expressed as absolute difference (the default level)
  - use `relevel()` to set another level of a categorical variable as default

Deviance

- For the “null” model and for the current model:

```
    Null deviance: 1453.6  on 1175  degrees of freedom
Residual deviance: 1284.3  on 1168  degrees of freedom
```

- Difference in deviance (169.3) is much higher than difference in parameters (7), suggesting that the current model is significantly better than the null model

Comparing models

- Let us add a presentation by interaction term:

```
> interaction.glm<-glm(choice~sex+age+presentation+
  product+sex:presentation,family="binomial")
```

- Are the extra-parameters justified?

```
> anova(sex_age_pres_prod.glm,interaction.glm,
  test="Chisq")
...
  Resid. Df Resid. Dev   Df Deviance P(>|Chi|)
1      1168    1284.25
2      1166    1277.68    2     6.57      0.04
```

- Apparently, yes (although `summary(interaction.glm)` suggests just a marginal interaction between sex and the percentage dummy variable)

Error rate

- The model makes an error when it assigns $p > .5$ to observation where `choice` is N or $p < .5$ to observation where `choice` is Y:

```
> sum((fitted(sex_age_pres_prod.glm)>.5 & choice=="N") |
     (fitted(sex_age_pres_prod.glm)<.5 & choice=="Y")) /
     length(choice)
[1] 0.2721088
```

- Compare to error rate by baseline model that always guesses the majority choice:

```
> table(choice)
choice
  N   Y
363 813
> sum(choice=="N")/length(choice)
[1] 0.3086735
```

- Improvement in error rate is nothing to write home about…

Binned fit

- Function from `languageR` package for plotting binned expected and observed proportions of 1-responses, as well as bootstrap validation, require logistic model fitted with `lrm()`, the logistic regression fitting function from the `Design` package:

```
> sex_age_pres_prod.glm<-
  lrm(choice~sex+age+presentation+product,
  x=TRUE,y=TRUE)
```

- The `languageR` version of the binned plot function (`plot.logistic.fit.fnc`) dies on our model, since it never predicts $p < 0.1$, so I hacked my own version, that you can find in the `r-data-1` directory:

```
> source("hacked.plot.logistic.fit.fnc.R")
> hacked.plot.logistic.fit.fnc(sex_age_pres_prod.glm,e)
```

- (Incidentally: in cases like this where something goes wrong, you can peek inside the function simply by typing its name)

Bootstrap estimation

- Validation using the logistic model estimated by `lrm()` and 1,000 iterations:

```
> validate(sex_age_pres_prod.glm,B=1000)
```

- When fed a logistic model, `validate()` returns various measures of fit we have not discussed: see, e.g., Baayen's book
- Independently of the interpretation of the measures, the size of the optimism indices gives a general idea of the amount of overfitting (not dramatic in this case)

Mixed model logistic regression

- You can use the `lmer()` function with the `family="binomial"` option
- E.g., introducing subjects as random effects:

```
> sex_age_pres_prod.lmer<-
  lmer(choice~sex+age+presentation+
  product+(1|subj),family="binomial")
```

- You can replicate most of the analyses illustrated above with this model

A warning

- Confusingly, the `fitted()` function applied to a `glm` object returns probabilities, whereas if applied to a `lmer` object it returns odd ratios
- Thus, to measure error rate you'll have to do something like:

```
> probs<-exp(fitted(sex_age_pres_prod.lmer)) /
  (1 +exp(fitted(sex_age_pres_prod.lmer)))
> sum((probs>.5 & choice=="N") |
  (probs<.5 & choice=="Y")) /
  length(choice)
```

- NB: Apparently, `hacked.plot.logistic.fit.fnc` dies when applied to an `lmer` object, on some versions of R (or `lme4`, or whatever)
- Surprisingly, fit of model with random subject effect is worse than the one of model with fixed effects only

Outline

Logistic regression

Logistic regression in R

Preparing the data and fitting the model

Practice

Practice time

- Go back to Navarrete's et al.'s picture naming data (`cwcc.txt`)
- Recall that the response can be a time (naming latency) in milliseconds, but also an error
- Are the errors randomly distributed, or can they be predicted from the same factors that determine latencies?
- We found a negative effect of repetition and a positive effect of position-within-category on naming latencies – are these factors also leading to less and more errors, respectively?

Practice time

- Construct a binary variable from responses (error vs. any response)
  - Use `sapply()`, and make sure that R understands this is a categorical variable with `as.factor()`
  - Add the resulting variable to your data-frame, e.g., if you called the data-frame `d` and the binary response variable `temp`, do:

    ```
    d$errorresp<-temp
    ```
  - This will make your life easier later on
- Analyze this new dependent variable using logistic regression (both with and without random effects)