Logit and Probit Models

Statistics and Econometrics

Jiahua Wu

Example 17.1 (slides 17-18)

The function to estimate logit and profit models is *glm*(). When we estimate logit (probit), we need specify *family* = "*binomial*" (as the response follows a Bernoulli distribution, which is a special case of binomial distribution), and *link* = "*logit*" for logit model and *link* = "*probit*" for probit model.

```
# Example 17.1
load("mroz.RData")

# estimation of binary response models
inlf.lpm <- glm(inlf ~ nwifeinc + educ + exper + expersq + age
                + kidslt6 + kidsge6, family = "gaussian", data)

inlf.lpm.lm <- lm(inlf ~ nwifeinc + educ + exper + expersq + age
                  + kidslt6 + kidsge6, data)

inlf.probit <- glm(inlf ~ nwifeinc + educ + exper + expersq + age
                   + kidslt6 + kidsge6, family = "binomial"(link = "probit"), data)

inlf.logit <- glm(inlf ~ nwifeinc + educ + exper + expersq + age
                  + kidslt6 + kidsge6, family = "binomial"(link = "logit"), data)
stargazer(inlf.lpm.lm, inlf.lpm, inlf.probit, inlf.logit, header = FALSE, type = 'latex',
          title = "Example 17.1. Labour Force Participation")
```

```
summary(inlf.logit)
```

```
##
## Call:
## glm(formula = inlf ~ nwifeinc + educ + exper + expersq + age +
##     kidslt6 + kidsge6, family = binomial(link = "logit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1770  -0.9063   0.4473   0.8561   2.4032
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.425452   0.860365   0.495  0.62095
## nwifeinc    -0.021345   0.008421  -2.535  0.01126 *
## educ         0.221170   0.043439   5.091 3.55e-07 ***
## exper        0.205870   0.032057   6.422 1.34e-10 ***
## expersq     -0.003154   0.001016  -3.104  0.00191 **
## age         -0.088024   0.014573  -6.040 1.54e-09 ***
## kidslt6     -1.443354   0.203583  -7.090 1.34e-12 ***
## kidsge6      0.060112   0.074789   0.804  0.42154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Table 1: Example 17.1. Labour Force Participation

	Dependent variable:			
		inlf		
	OLS	*normal*	*probit*	*logistic*
	(1)	(2)	(3)	(4)
nwifeinc	-0.003^{**}	-0.003^{**}	-0.012^{**}	-0.021^{**}
	(0.001)	(0.001)	(0.005)	(0.008)
educ	0.038^{***}	0.038^{***}	0.131^{***}	0.221^{***}
	(0.007)	(0.007)	(0.025)	(0.043)
exper	0.039^{***}	0.039^{***}	0.123^{***}	0.206^{***}
	(0.006)	(0.006)	(0.019)	(0.032)
expersq	-0.001^{***}	-0.001^{***}	-0.002^{***}	-0.003^{***}
	(0.0002)	(0.0002)	(0.001)	(0.001)
age	-0.016^{***}	-0.016^{***}	-0.053^{***}	-0.088^{***}
	(0.002)	(0.002)	(0.008)	(0.015)
kidslt6	-0.262^{***}	-0.262^{***}	-0.868^{***}	-1.443^{***}
	(0.034)	(0.034)	(0.118)	(0.204)
kidsge6	0.013	0.013	0.036	0.060
	(0.013)	(0.013)	(0.044)	(0.075)
Constant	0.586^{***}	0.586^{***}	0.270	0.425
	(0.154)	(0.154)	(0.508)	(0.860)
Observations	753	753	753	753
R^2	0.264			
Adjusted R^2	0.257			
Log Likelihood		−424.892	−401.302	−401.765
Akaike Inf. Crit.		865.785	818.604	819.530
Residual Std. Error	0.427 (df = 745)			
F Statistic	38.218^{***} (df = 7; 745)			
Note:				*p<0.1; **p<0.05; ***p<0.01

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  803.53  on 745  degrees of freedom
## AIC: 819.53
##
## Number of Fisher Scoring iterations: 4
```

First, let us try to understand the output of *glm*(). We take the logit model as an example. The first thing we observe from the summary is deviance residuals. There are five different types of residuals from *glm* models (see details: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/glm.summaries.html). The most commonly used one, and the one reported in summary by default is deviance residuals. The sum of squared of deviance residuals equals to the deviance of the estimated model. We can verify this by coding it manually as follows.

```
# deviance = sum of squared deviance residuals
summary(residuals(inlf.logit, type = "deviance"))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.17705 -0.90628  0.44733  0.03996  0.85606  2.40325
```

```
sum(residuals(inlf.logit, type = "deviance")^2)
```

```
## [1] 803.5303
```

```
inlf.logit$deviance
```

```
## [1] 803.5303
```

```
# deviance = -2 * log likelihood
(-2) * logLik(inlf.logit)
```

```
## 'log Lik.' 803.5303 (df=8)
```

The deviance of the model itself equals to -2 times the log-likelihood of the model. We can think of deviance as the couterpart of sum of squared residuals - we want to minimize sum of squared residuals in regression models, whereas deivance is minimized for logit and profit models (equivalent to maximizing log likelihood). The deviance residual is one way to measure the fit of each observation, just like residuals in multiple regression.

The interpretation of coefficients, standard errors and etc. is similar to that in linear regression models. One slight difference is that, instead of calling it t test, we refer to it as z test, as now with logit and probit models, the test statistic follows a standard normal distribution.

Dispersion parameter is not really useful for logit/probit models. It is mainly used in Poisson models to allow for different mean and variance (theoretically, mean and variance is always the same for Poisson distributed random variables). In practice, for any count data, we can well observe that mean and variance is different, and this dispersion parameter allows for a more flexbile model to fit the data.

Lastly, the maximum likelihood estimates are estimated by solving an optimization problem. Under the hood, it relies on an iterative algorithm (Newton-Raphson algorithm) to solve the problem. "Number of Fisher Scoring iterations" indicates the number of iterations required before convergence.

In the class, we discussed several different measures of goodness of fit. The following code shows how to calculate each of them.

```
### goodness of fit
# pseudo r-squared
1 - inlf.logit$deviance/inlf.logit$null.deviance
```

```
## [1] 0.2196814
```

```
# information criteria
AIC(inlf.logit)
```

```
## [1] 819.5303
```

```
BIC(inlf.logit)
```

```
## [1] 856.5228
```

The confusion matrix is a common tool for evaluating prediction performance of logit and profit models. We can use the function "confusionMatrix" from *caret* package to create confusion matrix. The interpretation of confusion matrix is discussed in the slides 13-14. Based on confusion matrix, we can calculate various metrics that can be used to compare different models. Three measures we discussed in the class are "Accuracy", "Sensitivity" (i.e., recall) and "Pos Pred Value" (i.e., precision).

```
inlf.predicted <- ifelse(inlf.logit$fitted.values < 0.5, 0, 1)
inlf.predicted <- as.factor(inlf.predicted)
confusionMatrix(inlf.predicted, as.factor(data$inlf), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 207  81
##          1 118 347
##
##                Accuracy : 0.7357
##                  95% CI : (0.7027, 0.7669)
##     No Information Rate : 0.5684
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.4539
##
##  Mcnemar's Test P-Value : 0.01071
##
##             Sensitivity : 0.8107
##             Specificity : 0.6369
##          Pos Pred Value : 0.7462
##          Neg Pred Value : 0.7188
##              Prevalence : 0.5684
##          Detection Rate : 0.4608
##    Detection Prevalence : 0.6175
##       Balanced Accuracy : 0.7238
##
##        'Positive' Class : 1
##
```

When it comes to hypothesis tesing, there are two functions that we can use, one is *lrtest*() from the *lmtest* package, and the other one is *linearHypothesis*() from the *car* package. Both of them are based on the likelihood ratio test. We reject null when test statistic is sufficiently large (p value is sufficiently small).

```
# test for overall significance
lrtest(inlf.logit)
```

```
## Likelihood ratio test
##
## Model 1: inlf ~ nwifeinc + educ + exper + expersq + age + kidslt6 + kidsge6
```

```
## Model 2: inlf ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   8 -401.77
## 2   1 -514.87 -7 226.22  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# test of overall significance manually
1 - pchisq(inlf.logit$null.deviance - inlf.logit$deviance,
           df = inlf.logit$df.null - inlf.logit$df.residual)
```

```
## [1] 0
```

```
# hypothesis testing
linearHypothesis(inlf.logit, c("exper = 0", "expersq = 0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## exper = 0
## expersq = 0
##
## Model 1: restricted model
## Model 2: inlf ~ nwifeinc + educ + exper + expersq + age + kidslt6 + kidsge6
##
##   Res.Df Df  Chisq Pr(>Chisq)
## 1    747
## 2    745  2 87.688  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Similar to multiple regression models, we can use *confint*() for confidence intervals. By default, the function produces 95% confidence intervals for all model coefficients.

```
# confidence intervals
confint(inlf.logit)
```

```
##                    2.5 %       97.5 %
## (Intercept) -1.262369605  2.115143766
## nwifeinc    -0.038174772 -0.005093629
## educ         0.137475679  0.308033451
## exper        0.143253654  0.269477613
## expersq     -0.005142506 -0.001110047
## age         -0.117137066 -0.059938042
## kidslt6     -1.853313005 -1.053969755
## kidsge6     -0.086032688  0.207521083
```

step() function still works for probit and logit, and the interpretation of output is exactly the same as that in linear regressions.

```
### model selection
inlf.null <- glm(inlf ~ 1, family = "binomial"(link = "logit"), data)
inlf.full <- glm(inlf ~ nwifeinc + educ + exper + expersq + age
                 + kidslt6 + kidsge6, family = "binomial"(link = "logit"), data)
step(inlf.null, scope = list(lower = inlf.null, upper = inlf.full), direction = "forward")
step(inlf.full, direction = "backward")
```

predict() function comes with two types, link or response. When *type* = "*link*", it returns the linear

combination of independent variables, i.e., $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_k x_k$. Alternatively, when $type =$ "*response*", it returns the predicted probability of $y = 1$. We can verify it by coding manually.

```
### prediction
new.ob = data.frame(nwifeinc = 10.91, educ = 12, exper = 14, expersq = 14^2,
                    age = 32, kidslt6 = 1, kidsge6 = 0)
predict(inlf.logit, newdata = new.ob, type = "link")
```

```
##         1
## 0.8504559
```

```
predict(inlf.logit, newdata = new.ob, type = "response")
```

```
##         1
## 0.7006628
```

```
# manual verification
xb <- sum(cbind(1, new.ob) * inlf.logit$coefficients); xb
```

```
## [1] 0.8504559
```

```
phat <- exp(xb)/(1 + exp(xb)); phat
```

```
## [1] 0.7006628
```

Lastly, let us discuss the interpretation of model. In the logit model, an estimated coefficient no longer measures the partial effect of the corresponding variable, as the partial effect of any independent variable now depends on values of all other independent variables (slide 9). In particular, for logit model, we can show that $g(z) = G(z) \cdot [1 - G(z)]$. Consequently, we can calculate average partial effect for x_j in a logit model using the formula

$$n^{-1} \sum_{i=1}^{n} G(\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik}) \cdot [1 - G(\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik})] \hat{\beta}_j,$$

where $G(z) = \frac{\exp(z)}{1+\exp(z)}$. The code to calcuate the average partial effect is given below.

```
# partial effects in logit model
mean(inlf.logit$fitted.values * (1 - inlf.logit$fitted.values)) * inlf.logit$coefficients
```

```
##   (Intercept)      nwifeinc          educ         exper       expersq
##  0.0759771297 -0.0038118135  0.0394965238  0.0367641056 -0.0005632587
##           age       kidslt6       kidsge6
## -0.0157193606 -0.2577536551  0.0107348186
```

If we compare them against estimated coefficients from the linear probability model (Table 1), they are quite similar. This is common in practice - coeffients from the linear probablity model usually give reliable estimates of average partial effects of independent variables in the model.

One key difference between LPM and logit/probit is that the marginal effect is assumed to be constant with LPM, however, it is no longer the case for logit/probit. That is, with LPM, an extra kid always decreases the chance of labor force participation by 0.26, regardless how many kids the individual already has. However, intuitively, we would think that the first kid has the greatest effect, and the marginal impact of an extra kid shall decrease. This nonlinearity is reflected in logit/probit output. We can calculate the impact of having the first kid less than 6, and the second kid less than 6, using the code below. It turns out the first kid reduces the probability of being in the labor force by 0.35, while the effect of the second kid is around 0.22. This nonlinearity allows logit/probit fit the data better, as reflected by log-likelihoods shown in Table 1.

```
# non-linear partial effects
new.ob <- with(data, data.frame(nwifeinc = mean(nwifeinc), educ = mean(educ), exper = mean(exper),
                    expersq = mean(expersq), age = mean(age), kidslt6 = c(0, 1, 2), kidsge6 = 1))
```

```r
# partial effect of having the first kid less than 6
predict(inlf.logit, newdata = new.ob[2, ], type = "response") -
        predict(inlf.logit, newdata = new.ob[1, ], type = "response")
```

```
##          2
## -0.3456246
```

```r
# partial effect of having the second kid less than 6
predict(inlf.logit, newdata = new.ob[3, ], type = "response") -
        predict(inlf.logit, newdata = new.ob[2, ], type = "response")
```

```
##          3
## -0.2157151
```