Day 3: Generalized Linear Regression

Kenneth Benoit and Slava Mikhaylov

Introduction to Data Science and Big Data Analytics

19 August 2015

Day 4 Outline

Classification models

- Binary Dependent Variables
- OLS vs logit

GAMs

- Extending linear regression

Classification

Overview

- Limited dependent variables (LDVs) refer to outcome variables that have finite, truncated, or discrete support:
  - Binary outcomes
  - Multichotomous outcomes (e.g. vote choice in multiparty elections)
  - Censored or truncated outcomes (e.g. salaries, counts).
- Longstanding tradition is to use linear models (e.g. OLS) for continuous outcomes and "non-linear" models for LDVs:
  - Logits or probits for binary, multichotomous, or ordered outcomes.
  - Poisson or negative binomial for counts.
  - Cox or parametric duration models for censored durations.

Binary Dependent Variables

- The linear probability model, which can be written as $Pr(y = 1|X) = \beta_0 + \beta_1 X$
- A drawback to the linear probability model is that predicted values are not constrained to be between 0 and 1 (unless the model includes only dummies, in which case it is inherently linear).
- An alternative is to model the probability as a function, $G(\beta_0 + \beta_1 X)$, where $0 < G(z) < 1$

The Logit Model

- One choice for $G(z)$ is the logistic function, which is the cdf for a standard logistic random variable

$$G(z) = \frac{exp(z)}{1 + exp(z)} = F(z)$$

- This case is referred to as a logit model, or sometimes as a logistic regression

The Probit Model

- Another popular choice for $G(z)$ is the standard normal cumulative distribution function (cdf)
- $G(z) = \Phi(z) \equiv \int \phi(v)dv$, where $\phi(z)$ is the standard normal, so

$$\phi(z) = \frac{1}{\sqrt{2\pi}} exp(-\frac{z^2}{2})$$

- This case is referred to as a probit model
- Both logit and probit have similar shapes – they are increasing in z, most quickly around 0
- Since this is a nonlinear model, it cannot be estimated by our usual methods
- Use maximum likelihood estimation

Maximum likelihood

- Developed by Ronald Fisher (UCL) in the 1920s.
- The estimate is the value of the parameter for which the observed data would have had greater chance of occurring than if the parameter equaled any other number

Maximum likelihood

- The "likelihood" is defined by the joint probability of the observed outcomes conditional on the data and β:

$$\begin{aligned}
\mathcal{L}(\beta|Y,\mathbf{X}) \equiv & \quad Pr[Y_1,...,Y_N|X_1,...,X_n;\beta] \\
= & \quad \prod_{i:Y_i=1} \Pr[Y_i=1|X_i] \prod_{i:Y_i=0}(1-\Pr[Y_i=1|X_i]) \\
= & \quad \prod_{i=1}^{N} \Lambda(X_i'\beta)^{Y_i}[1-\Lambda(X_i'\beta)]^{1-Y_i}
\end{aligned}$$

- MLE searches over possible βs until we maximize $\mathcal{L}(\beta|Y,\mathbf{X})$
- We can stabilize the computation by using $\log\left[\mathcal{L}(\beta|Y,\mathbf{X})\right]$.
- Deriving likelihood for probit swaps in $\Phi(.)$ for $\Lambda(X_i'\beta)$

- MLE estimates have some nice properties.
- If distributional assumptions are correct, MLE will be the most efficient way to estimate the model.
- Parameters are asymptotically normal.
- Under iid, standard errors are easy to compute.
- If iid is violated (e.g. clustering or heteroskedasticity), we can compute “robust” standard errors.
- (Note: OLS *is* the MLE solution for a linear model with normal errors.)

Logistic regression

The glm() function can be used to fit a logistic regression model by specifying "family=binomial".

```
library(ISLR)
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Sm
summary(glm.fit)
```

```
##
## 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
```

Logistic regression

Similar to linear models estimated with the lm(), the logistic regression model fitted with glm() can be examined with the summary() and coef().

```
coef(glm.fit)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938
##         Lag5       Volume
##  0.010313068  0.135440659
```

```
summary(glm.fit)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
summary(glm.fit)$coef[, 4]
```

Treatment effects for binary outcomes

- Comparing OLS with MLE logistic regression in estimating treatment effects.
- Most of the existing results indicate that there are differences but extremely slight (e.g. MHE on LPM)
- Since none of the models are exactly "right" (we are approximating the CEF), there are no real differences in terms of efficiency.
- But gains in interpretability, and estimation flexibility (e.g. IVs).

Treatment effects for binary outcomes

- Some issues with fixed effects estimation.
- In short panels, joint estimation of FEs and model parameters results in inconsistent estimation of all parameters.
- The inconsistent estimation of FEs propagates to inconsistent estimation of ρ and β more generally.
- An MLE solution is a "conditional logit" estimator. But not available for probit. (For more see Neil Beck (2011) "Is OLS with a binary dependent variable really OK?: Estimating (mostly) TSCS models with binary dependent variables and fixed effects.")
- Even if the parameters can be consistently estimated, it may not be possible to consistently estimate the marginal effects (especially for multiplicative effects).
- OLS doesn't suffer from these issues, although in such setting (binary treatment and FEs) it's still difficult to get good identification on the causal effect.

One perspective (Angrist 2001, p.3):

[T]echnical challenges posed by LDV models come primarily from what I see as a counterproductive focus on structural parameters such as a latent index coefficients or censored regression coefficients instead of directly interpretable causal effects. In my view, the problem of causal inference with LDV's is not fundamentally different from causal inference with continuous outcomes.

- Angrist proposes that OLS and related methods (quantile regression, Abadie-type kappa weighting) produce causal effect estimates either identical or more reliable than those from non-linear models.

But others disagree (Imbens 2001, 18-20)

[T]he choice of the estimand is distinct from the statistical question of the specification of the model...The aim is to provide a flexible approximation. [F]or a binomial distribution the logistic regression model can be thought of as providing a linear approximation to the log odds ratio, this choice is...an appealing one...In cases with other limited dependent variables, alternative nonlinear models may be appropriate.

Remarks

- From reading applied literature, it appears that it is ok to use non-linear MLE estimators of treatment effects so long as they are interpreted correctly.
- Non-linear MLE estimators tend not to differ very much from linear OLS estimators, in applied settings.
- It may be, that problems with fixed effects in non-linear models, for example, tend to dominate over problems of non-sensical predictions from linear models. However, there are certainly cases when the opposite would hold.

GAMs

Polynomial regression

- Standard linear regression

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

- Easy extension to nonlinearity (we've seen it already) with a polynomial regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \epsilon_i$$

Polynomial regression

```
library(ISLR)
fit <- lm(wage ~ poly(age, 4), data = Wage)
coef(summary(fit))
```

```
##                   Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)      111.70361  0.7287409 153.283015 0.000000e+00
## poly(age, 4)1    447.06785 39.9147851  11.200558 1.484604e-28
## poly(age, 4)2   -478.31581 39.9147851 -11.983424 2.355831e-32
## poly(age, 4)3    125.52169 39.9147851   3.144742 1.678622e-03
## poly(age, 4)4    -77.91118 39.9147851  -1.951938 5.103865e-02
```

Polynomial regression

We can specify them directly

```
fit2a <- lm(wage ~ age + I(age^2) + I(age^3) + I(age^4), data = Wage)
coef(fit2a)
```

```
##   (Intercept)           age      I(age^2)      I(age^3)      I(age^4)
## -1.841542e+02  2.124552e+01 -5.638593e-01  6.810688e-03 -3.203830e-05
```

Or use cbind() function:

```
fit2b <- lm(wage ~ cbind(age, age^2, age^3, age^4), data = Wage)
```

Polynomial regression

```
fit.1 <- lm(wage ~ age, data = Wage)
fit.2 <- lm(wage ~ poly(age, 2), data = Wage)
fit.3 <- lm(wage ~ poly(age, 3), data = Wage)
fit.4 <- lm(wage ~ poly(age, 4), data = Wage)
fit.5 <- lm(wage ~ poly(age, 5), data = Wage)
anova(fit.1, fit.2, fit.3, fit.4, fit.5)


## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8888  0.001679 **
## 4   2995 4771604  1      6070   3.8098  0.051046 .
## 5   2994 4770322  1      1283   0.8050  0.369682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Polynomial regression

```
fit <- glm(I(wage > 250) ~ poly(age, 4), data = Wage, family = binomial)
summary(fit)
```

```
##
## Call:
## glm(formula = I(wage > 250) ~ poly(age, 4), family = binomial,
##     data = Wage)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3110  -0.2607  -0.2488  -0.1791   3.7859
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.3012     0.3451 -12.465  < 2e-16 ***
## poly(age, 4)1  71.9642    26.1176   2.755  0.00586 **
## poly(age, 4)2 -85.7729    35.9043  -2.389  0.01690 *
## poly(age, 4)3  34.1626    19.6890   1.735  0.08272 .
## poly(age, 4)4 -47.4008    24.0909  -1.968  0.04912 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 730.53  on 2999  degrees of freedom
```

Basis functions

- Polynomial regression is a special case of a basis function:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \cdots + \beta_K b_K(x_i) + \epsilon_i$$

- We choose basis function beforehand, i.e. they are fixed and known and we don't need to estimate them.
- E.g. for a polynomial regression basis functions are $b_j(x_i) = x_i^j$.
- It's a standard additive model and we can use standard tools like linear regression to estimate parameters in this model.

Regression splines

- We don't need to fit a polynomial regression over the entire range of X. For more flexible fit we can fit lower degree polynomials over different regions of X.
- Points where coefficients change are called knots.

Regression splines

We first use bs() to generate a basis matrix for a polynomial spline and fit a model with knots at age 25, 40 and 60.

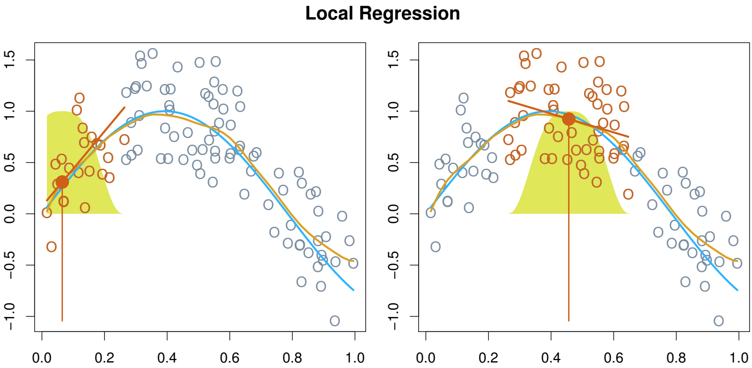
```
library(splines)
fit <- lm(wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)
```

Alternatively, the df() function can be used to produce a spline fit with knots at uniform intervals.

```
fit2 <- lm(wage ~ ns(age, df = 4), data = Wage)
```

Local regression

- Local regression fits a weighted linear regression at a local point x_0 using its neighborhood as training set.



Local regression: Algorithm

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.
3. Fit a weighted least squares regression of the y_i on the x_i using the above weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$

Local regression

```
fit <- loess(wage ~ age, span = 0.2, data = Wage)
```

The most important choice in fitting a local regression is the span s defined in Step 1. It controls the flexibility of your fit.

```
fit2 <- loess(wage ~ age, span = 0.5, data = Wage)
```

GAMs

- Generalized Additive Models take all previous extensions into the multiple regression domain:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

- GAMs allow non-linear functions of the variables while maintaining additivity:

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

We calculate separate f_i for each X_j and then add together all individual contributions.

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i$$

Most of the methods we mentioned before can be used to build individual basis functions.

GAMs

- We can fit a GAM with lm() when an appropriate basis function can be used. Here lm() allows using natural splines function ns() from the base "splines" package.

```
gam1 <- lm(wage ~ ns(year, 4) + ns(age, 5) + education, data = Wage)
```

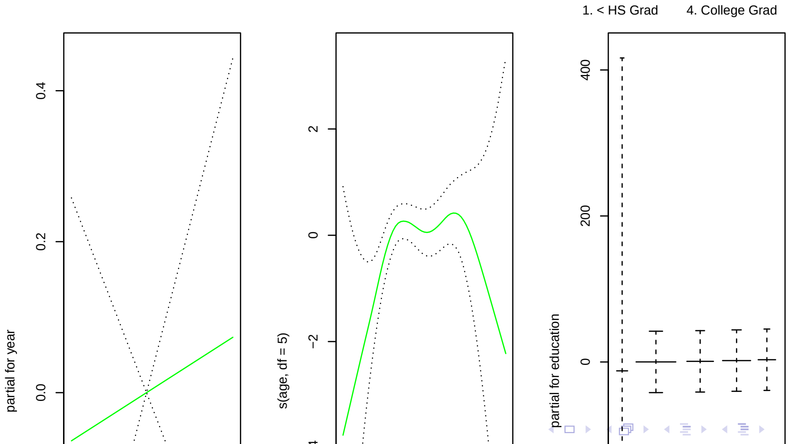
- A general solution to fitting GAMs is offered in "gam" package. This is especially useful when splines cannot be easily expressed in terms of basis functions.

```
library(gam)

## Loaded gam 1.09.1

gam.m3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
```

GAMs for classification

The gam() function also allows fitting logistic regression GAM with the 'family = binomial' argument.

```
gam.lr <- gam(I(wage > 250) ~ year + s(age, df = 5) + education, family = binom
par(mfrow = c(1, 3))
plot(gam.lr, se = T, col = "green")
```



GAMs: benefits and limitations

- With GAMs we estimate non-linear relationships but maintain additivity (identify effect of each X_j on Y individually).
- The drawback is also that the model is additive. We can manually fit some interactions there. But that's largely limited to small number of predictors, i.e. low-dimensional.