Part 9 Linear model extensions

Dr. Nguyen Quang Huy

July 20, 2020

Overview

In this section, we will discuss about the following topic:

1. Regression splines
   - Piecewise polynomials
   - Constraints and splines
   - Discussion on the number and the locations of the knots
2. Smoothing splines
   - Introduction to smoothing splines
   - Choosing the smoothing parameter
3. Local regression
4. Generalized additive models (GAM)
5. Generalized linear models (GLM)

Regression splines

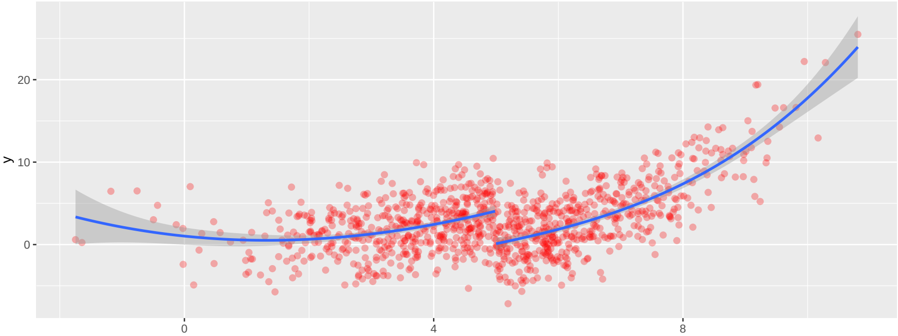
- Instead of fitting a high-degree polynomial over the entire range of **X**, piecewise polynomial regression fits separate low-degree polynomials over different regions of **X** (splines)
- A piecewise cubic polynomial with 1 knot at points k_1 takes the form

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x < k_1 \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } k_1 \leq x \end{cases}$$

- In general, if we place J different knots throughout the range of X, then we will fit $(J+1)$ different polynomials.
- **Problem**: the function is discontinuous at knots.

Regression splines

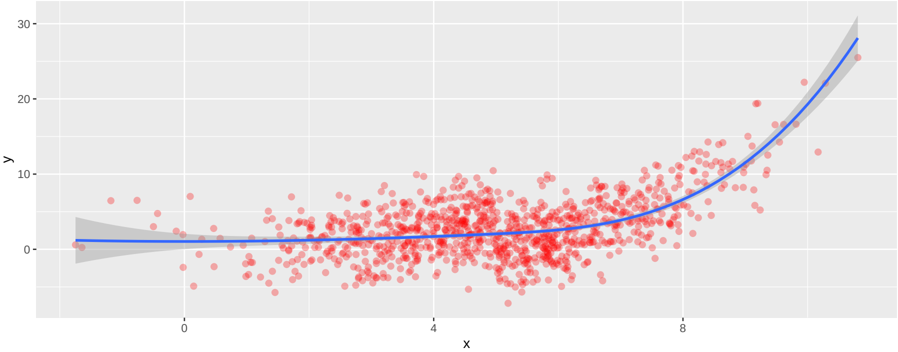
```
x<-rnorm(1000,5,2)
y=ifelse(x<5,0.03*x^3-0.2*x+1,0.02*x^3-0.1*x-2)+rnorm(1000,0,3
z<-ifelse(x<5,TRUE,FALSE)
dat<-data.frame(x,y,z)
dat%>%ggplot(aes(x,y,group=z))+geom_point(col="red",cex=2,alph
  geom_smooth(method=lm,formula=y~poly(x,3,raw=TRUE))
```



Regression splines

To solve the uncontinuous problem, they add two additional constraints: both the first and second derivatives of the piecewise polynomials are continuous.

```
dat%>%ggplot(aes(x,y))+geom_point(aes(group=z),col="red",cex=2
  geom_smooth(method=lm,formula=y~bs(x,knots=5))
```



Regression splines

We can prove that, the estimation of a **continuous cubic splines** with m knots, k_1, k_2, $\cdots$, k_m is similar to the esimation of

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 h_1(x_i) + \beta_5 h_2(x_i) + \cdots + \beta_{m+3} h_m(x_i) + \epsilon_i$$

where

$$h_j(x_i) = \begin{cases} (x - k_j)^3 & \text{if } x \geq k_j \\ 0 & \text{if } x < k_j \end{cases}$$

Where should we place the knots

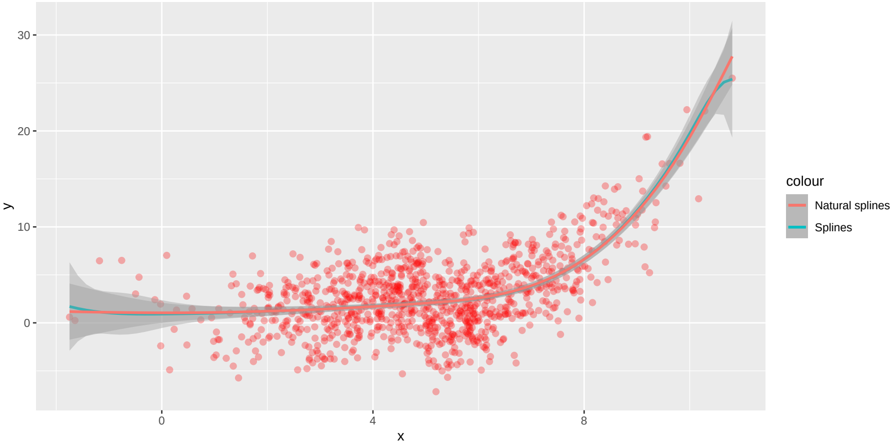
- Manually: place more knots in places where the function might vary most rapidly, and to place fewer knots where it seems more stable.
- Automatically: specify the desired degrees of freedom (number of β), and have the software automatically place the corresponding number of knots (cross validation)

Regression splines

- A (continuous) cubic splines has $(m+3)$ parameters where m is the number of knots
- Splines can have high variance at the outer range of the predictors, that is, when X takes on either a very small or very large value
- They introduce "natural spline" which is a spline with additional boundary constraints: the function is required to be **linear** at the boundary
  - In the region where X is spline smaller than the smallest knot
  - or in the region where X larger than the largest knot
- A natural cubic splines has m parameteres where m is the number of knows

Regression splines

```
dat%>%ggplot(aes(x,y))+geom_point(aes(group=z),col="red",cex=2
  geom_smooth(method=lm,formula=y~bs(x,knots=c(0,5,10)),aes(co
  geom_smooth(method=lm,formula=y~ns(x,knots=c(0,5,10)),aes(co
```



Smoothing splines

- What we do is find some function $f(x)$ that fits the observed data well i.e. minimize $RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2$.
- If we don't put any constraints on $f(x)$, then we can make RSS small simply by choosing f such that it interpolates all of the y_i, or it is easy to overfit
- What we want function f that makes RSS small, but f is also **smooth**.
- We find the function f that minimizes

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int f^{''}(x)\ dx$$

where $\lambda \int f^{''}(x)\ dx$ is a penalty term.

- The second derivative is the speed of change of the slope i.e. associated with the roughness of the curve.

Smoothing splines

- It can be shown that this problem has an explicit, finite-dimensional, which is a **natural cubic spline** with knots at the unique values of the x_i, $i = 1, ..., n$
- It seems that function f is over-parametrized (n degree of freedom)
  - Regression splines with n knots has $(n + 4)$ parameter
  - Linear constrains when $x < min(k_i)$ and $x > max(k_i) \rightarrow n$ parameter left
- The penalty term translates to a penalty on the spline coefficients, which are shrunk the model toward the linear fit.
- The solution has the following form, where b_i are basis functions

$$f(x) = \sum_{i=1}^{n} \beta_i b_i(x)$$

Smoothing splines

Solution of smoothing splines (forget it :D): for each λ, find vector β to minimize

$$(\mathbf{y} - \mathbf{B}\beta)^{'} (\mathbf{y} - \mathbf{B}\beta) + \lambda\beta^{'}\Omega\beta$$

where $B_{ij} = b_i(x_j)$ and $\Omega_{ij} = \int b_i^{''}(t) b_j^{''}(t)$. The solution is

$$\hat{\beta} = \left(B^{'}B + \lambda\Omega\right) B^{'}\mathbf{y}$$

Thus, the smoothing splines is

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\beta}_i b_i(x)$$

Let $\hat{\mathbf{f}} = (f(x_1), f(x_2), \cdots, f(x_n))$, we have

$$\hat{\mathbf{f}} = \mathbf{B}\hat{\beta} = \mathbf{B}\left(B^{'}B + \lambda\Omega\right) B^{'}\mathbf{y} = \mathbf{S}_\lambda \mathbf{y}$$

Smoothing splines

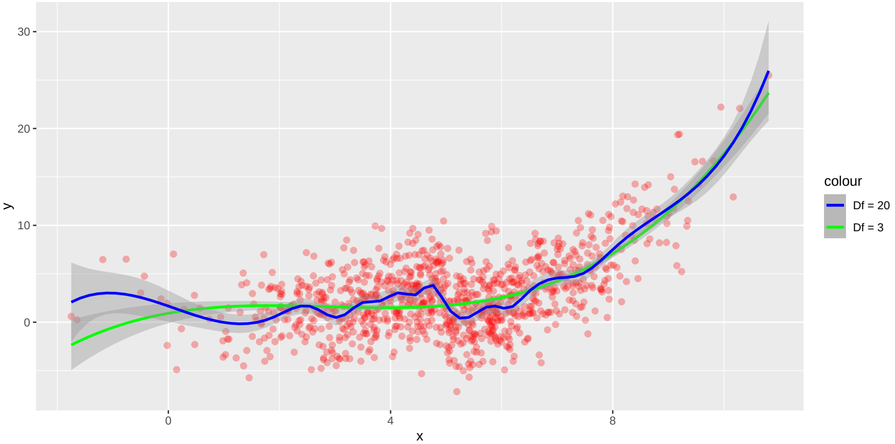
The effective degree of freedom is defined as “the sum of the sensitivities of the fitted values with respect to the observed response values”

$$\sum_{i=1}^{n} \frac{\partial \hat{y}_i}{\partial y_i} = trace(\mathbf{S}_\lambda)$$

- In a linear regression model with k predictors, the degree of freedom is the number of parameters: (k+1)
- In smoothing splines, there are n parameters, but they are contrained by others (because of λ) $\rightarrow$ the effective degree of freedom is $trace(\mathbf{S}_\lambda)$
- Larger effective degree of freedom, higher variance.
- Lower effective degree of freedom, lower variance.

Regression splines

```
dat%>%ggplot(aes(x,y))+geom_point(aes(group=z),col="red",cex=2
  geom_smooth(method=lm,formula=y~ splines::bs(x,df=3) ,aes(co
  geom_smooth(method=lm,formula=y~ splines::bs(x,df=20),aes(co
```



Local regression

Local regression is a different approach for fitting flexible non-linear funtions, which involves computing the fit at a target point using **only the regression nearby training observations.**

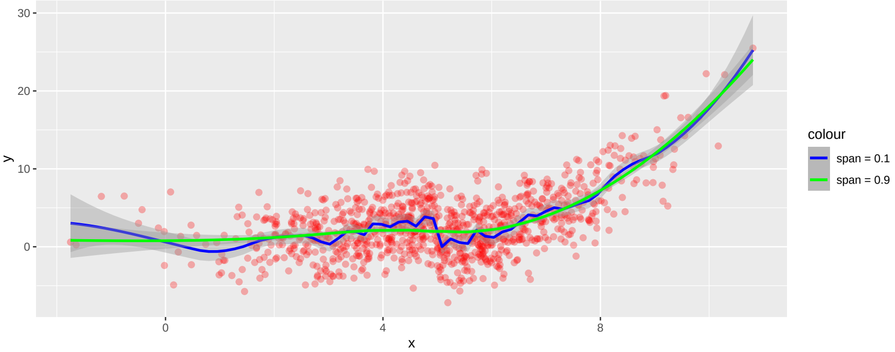
- Gather the fraction $s = k/n$ (span) of training points whose x_i are closest to x.
- Choose a weight function K to each point in this neighborhood so that the point furthest from x has weight zero, and the closest has the highest weight; $K_i = K(x_i, x)$
- Fit a weighted least squares regression: finding β_0, β_1 that minimize

$$\sum K_i (y_i - \beta_0 - \beta_1 x_i)^2$$

- Return $f(x) = \hat{\beta}_0 + \hat{\beta}_1 x$

Regression splines

```
dat%>%ggplot(aes(x,y))+geom_point(aes(group=z),col="red",cex=2
  geom_smooth(method="loess",span=0.1,aes(colour="span = 0.1")
  geom_smooth(method="loess",span=0.9,aes(colour="span = 0.9")
  scale_colour_manual(values=c("blue","green"))
```



Generalized additive models

- We have presented a number of approaches for flexibly predicting a response Y on the basis of a single predictor X
- **Generalized additive models (GAMs)** provide a general framework for extending a standard linear model by allowing non-linear functions of each of the variables.
- GAM can be write as

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

  where f_i can be a constant, a polinomal, a natural splines, or a smoothing splines ...
- When f_j is a smoothing splines, the least square method can not be used. GAM method fits a model involving multiple predictors by repeatedly updating the fit for each predictor in turn, that is, apply the fitting method for that variable to a *partial residual*

Generalized additive models

Pros of GAMs

1. GAMs allow us to automatically fit a non-linear f_j to each X_j, we do not need to manually try out many different transformations on each variable individually.
2. The non-linear function f_i can potentially make more accurate predictions.
3. We can examine the effect of each X_j on Y individually. If we are interested in inference, GAMs provide a useful representation.

Cons of GAMs

1. The main limitation of GAMs is that the model is restricted to be additive (linears). With many variables, important interactions can be missed.

Generalized additive models - practice

Load dataset **Boston** from **MASS** packages and build GAM to predict **medv**.

```
dat<-Boston
```

1. Standardize all numerical variables (except for **medv**) and split data into trainning set and test set (80%-20%)
2. Build a linear model when $medv \sim lstat$ (use polinomial, splines, natural splines, smoothing splines). Which model has the lowest error (on test dataset)?
3. Build a GAM where *medv* depends on all predictors.

Generalized additive models - practice

```
dat<-Boston
# STANDARDIZE
standardize<-function(x){x<-(x-mean(x,na.rm=TRUE))/sd(x,na.rm
for (col in names(dat)){
  if((col!="medv")&class(dat[,col]) %in% c("integer","numeric"
    dat[,col]<-standardize(dat[,col])
  }
}
# SPLITTING INTO TRAIN - TEST
set.seed(1)
test_index<-createDataPartition(dat$medv, times = 1, p = 0.2,l
train<-dat[-test_index,]
test<-dat[test_index,]
```

Generalized additive models - practice

```r
## POLINOMIAL
poly.fit<-lm(medv~poly(lstat,4,raw=TRUE),data=train)
medv.pred<-predict(poly.fit,test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 5.61816
```

```r
## CUBIC SPLINES (CONTINUOUS)
cubic.splines<-lm(medv~bs(lstat,knots = c(0)),data=train)
medv.pred<-predict(cubic.splines,test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 5.62075
```

Generalized additive models - practice

```r
library(gam)
# NATURAL SPLINE (LINEAR when x small and x large)
natural.splines<-lm(medv~ns(lstat,df=5),data=train)
medv.pred<-predict(natural.splines,test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 5.626698
```

```r
# SMOOTHING SPLINES (require gam package)
df1<-smooth.spline(train$medv,train$lstat,cv=TRUE)$df
smoothing.spl<-gam(medv~s(lstat,df=df1),data=train)
medv.pred<-predict(smoothing.spl, newdata = test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 5.606942
```

Generalized additive models - practice

```
# Different functions in GAM model
gam1<-gam(medv~s(lstat,6)+lo(dis,0.3)+
            poly(crim,4,raw=TRUE)+zn,data=train)
medv.pred<-predict(gam1, newdata = test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 5.055312
```

```
# ADDITIVE
gam2<-gam(medv~s(lstat)+s(dis)+
            s(crim)+s(zn),data=train)
medv.pred<-predict(gam2, newdata = test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 4.957187
```

Generalized additive models - practice

```
## MANUALLY CHOOSE MODEL :)
gam3<-gam(medv~s(lstat)+s(crim)+s(zn)+s(indus)+
            s(nox)+s(rm)+s(dis)+s(age)+s(tax)+
            s(ptratio)+chas, data=train)
medv.pred<-predict(gam3, newdata = test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 3.769918
```

```
gam4<-mgcv::gam(medv~s(lstat)+s(crim)+s(zn)+
            s(indus)+s(nox)+s(rm)+s(dis)+
            s(age)+s(tax)+s(ptratio)+chas+s(black)+rad,
          data=train,select=TRUE)
medv.pred<-predict(gam4, newdata = test)
sqrt(mean((medv.pred-test$medv)^2))
```

```
## [1] 3.299333
```

Generalized additive models

GAMs are generally fit using a *backfitting* approach.

```
n<-1000
set.seed(1)
x1<-rnorm(n,0,1)
x2<-rnorm(n,0,1)
x3<-rnorm(n,0,1)
y<-1+2*x1+3*x2+4*x3+rnorm(n,0,5)
lm(y~x1+x2+x3) # easy to perform a multi-regression
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Coefficients:
## (Intercept)           x1           x2           x3
##       1.085        2.115        3.076        4.046
```

Generalized additive models

Suppose that you only have a computer to perform simple linear regression.

1. Fit the model $Y \sim X_1$ to obtain $\hat{\beta}_1$.
2. Fit the model $Y - \hat{\beta}_1 X_1 \sim X_2$ to obtain $\hat{\beta}_2$.
3. Fit the model $Y - \hat{\beta}_1 X_1 - \hat{\beta}_2 X_2 \sim X_3$ to obtain $\hat{\beta}_3$.
4. Back to step (1), replace Y by $Y - \hat{\beta}_2 X_2 - \hat{\beta}_3 X_3$.
5. Write a for loop to repeat these steps N times ($N = 10$)
6. With $\hat{\beta}_1$, $\hat{\beta}_2$ and $\hat{\beta}_3$ from step (5), calculate $\hat{\beta}_0$ as follow

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}_1 - \hat{\beta}_2 \bar{X}_2 - \hat{\beta}_3 \bar{X}_3$$

Generalized additive models

```
N<-5
b<-matrix(0,3,N)
for(i in 2:N){
  b[1,i]<-lm(y-b[2,i-1]*x2-b[3,i-1]*x3~x1)$coef[2]
  b[2,i]<-lm(y-b[1,i-1]*x1-b[3,i-1]*x3~x2)$coef[2]
  b[3,i]<-lm(y-b[2,i-1]*x2-b[1,i-1]*x1~x3)$coef[2]
}
b[,N]
```

```
## [1] 2.114983 3.075799 4.046309
```

```
b[1,]
```

```
## [1] 0.000000 2.333841 2.105754 2.115697 2.114983
```

```
mean(y)-b[1,N]*mean(x1)-b[2,N]*mean(x2)-b[3,N]*mean(x3)
```

```
## [1] 1.085474
```

Generalized linear models (GLM)

We have the linear model

$$Y = \beta_1 X_1 + \cdots \beta_p X_p + \epsilon = \mathbf{X}^{'}\beta + \epsilon$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Or we can write

$$\mathbb{E}(Y|X) = \mathbf{X}^{'}\beta$$

where $Y|X$ has normal distribution function. Consider a more general case, where

$$g\left(\mathbb{E}(Y|X)\right) = \mathbf{X}^{'}\beta$$

where g is called link function and $Y|X$ has distribution function F. g must be monotone and differenciable.

Generalized linear models

- In linear model, we assume that $Y|X = x_i$ has normal distribution, but it is not appropriate in many situation.
- In credit risk modeling, Y is binary, $\rightarrow$ $Y|X = x_i$ is binary, we can not assume that $Y|X = x_i$ has normal distribution. A suitable distrbution for $Y|X = xi$ is Bernoulli distribution.
- When Y is a counting number, a number of claim in a year for example, $Y|X = x_i$ is a counting number. $Y|X = x_i$ can not have normal distribution. $Y|X = x_i$ can be a Poisson or a Negative Binomial random variable.
- Even when Y is continuous, but $Y > 0$, then $Y|X = x_i > 0 \forall x_i$; the normality assumption of $Y|X = x_i$ should be considered.

Generalized linear models

- In linear regression, we try to build model where $\mathbb{E}(Y|X = x_i)$ is a linear function of x_i i.e. find β_0 and β_1 such that $\mathbb{E}(Y|X = x_i) = \beta_0 + \beta_1 x_i$.
- $\beta_0 + \beta_1 x_i$ can take any value on $\mathbb{R}$
- $\mathbb{E}(Y|X = x_i)$, however, depends on the distribution of $Y|X$.
- If $Y|X = x_i$ has binomial distribution then $\mathbb{E}(Y|X = x_i) \in [0, 1]$.
- If $Y|X = x_i$ has poisson distribution then $\mathbb{E}(Y|X = x_i) \in [0, \infty)$.
- To match the range of $\mathbb{E}(Y|X = x_i)$ to set $\mathbb{R}$, they introduce a link function g.
    - For example, $\mathbb{E}(Y|X = x_i) \in [0, 1]$, we need function g such that $g : [0, 1] \to \mathbb{R}$
    - When $\mathbb{E}(Y|X = x_i) \in [0, \infty)$, we need function g such that $g : [0, \infty] \to \mathbb{R}$

Generalized linear models

- When $Y|X = x_i$ is a Bernoulli random variable, any function $g : [0, 1] \to \mathbb{R}$ can be a link function.
- However, when $Y|X = x_i \sim \mathcal{B}(p_i)$, the logit function is the canonical link function. The logit function is defined as follows

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

- Definition of canonical link function is out-of-scope of this course. All you should know is that when the link function g is the canonical link, we have a analytic solution for $\beta_0, \beta_1, \cdots$
- It explain why when $Y|X = x_i$ is a Bernoulli, they choose g as logit function. For these choice of Y and g, we often call **logistic regression**,

Generalized linear models

- Quick question 1: Name two other functions that can be a link function when $Y|X = x_i$ is a Bernoulli

Generalized linear models

- Quick question 1: Name two other functions that can be a link function when $Y|X = x_i$ is a Bernoulli
- $\Phi^{-1}(x)$: the inverse function of the distribution function of standard normal random variable (Probit regression)

Generalized linear models

- Quick question 1: Name two other functions that can be a link function when $Y|X = x_i$ is a Bernoulli
- $\Phi^{-1}(x)$: the inverse function of the distribution function of standard normal random variable (Probit regression)
- $log\left(-log(1-x)\right)$

Generalized linear models

- Quick question 1: Name two other functions that can be a link function when $Y|X = x_i$ is a Bernoulli
- $\Phi^{-1}(x)$: the inverse function of the distribution function of standard normal random variable (Probit regression)
- $log\left(-log(1-x)\right)$
- Quick question 2: Name a function that can be a link function when $Y|X = x_i$ is a Poisson

Generalized linear models

- Quick question 1: Name two other functions that can be a link function when $Y|X = x_i$ is a Bernoulli
- $\Phi^{-1}(x)$: the inverse function of the distribution function of standard normal random variable (Probit regression)
- $log\left(-log(1-x)\right)$
- Quick question 2: Name a function that can be a link function when $Y|X = x_i$ is a Poisson
- $log(x)$

Generalized linear models - estimation

- Suppose that $Y|(X = x_i)$ has density function/p.m.f f_i with parameter θ_i
- f_i is the density/p.m.f function of random variable $Y_i = Y|(X = x_i)$ where $g\left(\mathbb{E}(Y_i)\right) = \mathbf{x}_i'\beta$
- Because $\mathbb{E}(Y_i)$ is a function of θ_i, we have can write $\theta_i = h(\mathbf{x}_i'\beta)$
- We must find β to maximize the log likelyhood function

$$L(\beta) = Log \prod_{i=1}^{n} f_i(y_i) = \sum_{i=1}^{n} Log(f_i(y_i))$$

where paramater of f_i is $\theta_i = h(\mathbf{x}_i'\beta)$

Generalized linear models - estimation

- For example, when f_i is the p.m.f of Bernoulli random variable with parameter p_i and g is logit function: $g(x) = log(x/(1-x))$
- What is the p.m.f f_i at y_i: $f_i(y_i) = \mathbb{P}(Y_i = y_i))$

Generalized linear models - estimation

- For example, when f_i is the p.m.f of Bernoulli random variable with parameter p_i and g is logit function: $g(x) = log(x/(1-x))$
- What is the p.m.f f_i at y_i: $f_i(y_i) = \mathbb{P}(Y_i = y_i))$

$$f_i(y_i) = p_i^{y_i} \times (1-p_i)^{(1-y_i)} \rightarrow log(f_i) = y_i log(p_i) + (1-y_i) log(1-p_i)$$

- What is the relation between p_i and $\mathbf{x}_i^{'}\beta$

$$\begin{aligned} & g\left(\mathbb{E}(Y_i)\right) = \mathbf{x}_i^{'}\beta \\ \rightarrow \quad & log\left(\frac{p_i}{1-p_i}\right) = \mathbf{x}_i^{'}\beta \\ \rightarrow \quad & p_i = \frac{exp(\mathbf{x}_i^{'}\beta)}{1+exp(\mathbf{x}_i^{'}\beta)} \end{aligned}$$

Generalized linear models - estimation

- Write $\sum log(f_i)$ as a function of β

$$
\begin{aligned}
L(\beta) &= \sum_{i=1}^{n} log(f_i) \\
&= \sum_{i=1}^{n} y_i log\left[\frac{exp(\mathbf{x}_i'\beta)}{1+exp(\mathbf{x}_i'\beta)}\right] + (1-y_i) log\left[\frac{1}{1+exp(\mathbf{x}_i'\beta)}\right] \\
&= \sum_{i=1}^{n} y_i\mathbf{x}_i'\beta - log\left[1+exp(\mathbf{x}_i'\beta)\right]
\end{aligned}
$$

- Methods to solve for β
  - Newton–Raphson method
  - Iteratively reweighted least squares method

Generalized linear models

- Using GLM model to predict credit card default (data **Default** in **ISLR** package)

```
dat<-Default
head(Default)
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```
summary(Default$default)
```

```
##   No  Yes
## 9667  333
```

Generalized linear models

```
# STANDARDIZE
standardize<-function(x){x<-(x-mean(x,na.rm=TRUE))/sd(x,na.rm
for (col in names(dat)){
  if(class(dat[,col]) %in% c("integer","numeric")){
    dat[,col]<-standardize(dat[,col])
  }
}
# SPLITTING INTO TRAIN - TEST
set.seed(1)
test_index<-createDataPartition(dat$default, times = 1, p = 0.
train<-dat[-test_index,]
test<-dat[test_index,]
summary(train$default)
```

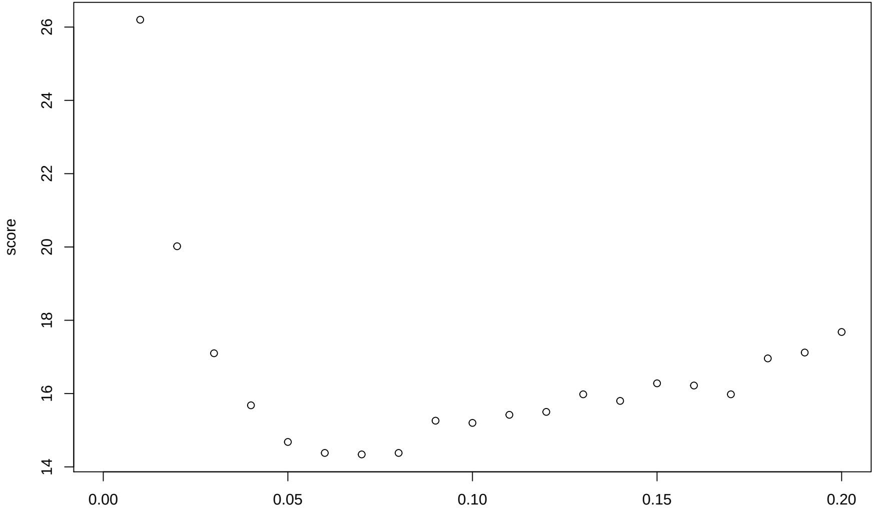
```
##   No  Yes
## 4833  166
```

Generalized linear models

```r
logit<-glm(default~.,
           family = binomial(link = "logit"),data=train)
pred<-predict(logit,test,type="response")
#Fixed cut point
log.pred1<-as.factor(ifelse(pred>0.1,"Yes","No"))
table(log.pred1,test$default)
```

```
##
## log.pred1   No  Yes
##        No 4563   45
##        Yes  271  122
```

Generalized linear models



Generalized linear models

```
finalcut<-cutpoint[which.min(score)]
logit<-glm(default~.,
           family = binomial(link = "logit"),data=train)
pred<-predict(logit,test,type="response")
log.pred1<-as.factor(ifelse(pred>finalcut,"Yes","No"))
table(log.pred1,test$default)
```

```
##
## log.pred1   No  Yes
##        No 4465   35
##       Yes  369  132
```

Generalized linear models

```
finalcut<-cutpoint[which.min(score)]
probit<-glm(default~.,
           family = binomial(link = "probit"),data=train)
pred<-predict(probit,test,type="response")
log.pred1<-as.factor(ifelse(pred>finalcut,"Yes","No"))
table(log.pred1,test$default)
```

```
##
## log.pred1    No  Yes
##       No   4414   31
##       Yes   420  136
```

GLM and GAM

- In the general GLM model, we have

$$g\left(\mathbb{E}(Y|X = x_i)\right) = \mathbf{x}_i^{'}\beta$$

- We could extend the right hand side of this formula by the sum of function of x_{ij}

$$g\left(\mathbb{E}(Y|X = x_i)\right) = \sum_{j=1}^{p} f_j(x_{ij})$$

where f_j can be a polinomial, natural splines, smoothing splines ...

GLM and GAM

```r
logit<-gam(default~ student + s(income) + s(balance) ,
           family = binomial,data=train)
pred<-predict(logit,test,type="response")
log.pred1<-as.factor(ifelse(pred>finalcut,"Yes","No"))
table(log.pred1,test$default)
```

```
##
## log.pred1   No  Yes
##       No  4434   34
##       Yes  400  133
```

Exercise

Exercise 1 In this exercise, we will analyze the **Wage** data set.

- Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial.
- Using natural splines and smoothing splines to predict wage using age.
- Make a plot of the resulting polynomial fit to the data.

Exercise 2 This exercise relates to the **College** data set.

- Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.
- Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.