I’ve shown how to estimate a standard Cumulative Logit model with the ordinal::clm function and its use case in credit risk models below.

Modelling LGD with Propotional Odds Model

In the real-world LGD data, we usually would observe 3 ordered categories of values, including 0, 1, and in-betweens. In cases with a nontrivial number of 0 and 1 values, the ordered logit model, which is also known as Proportional Odds model, can be applicable. In the demonstration below, I will show how we can potentially use the proportional odds model in the LGD model development.

First of all, we need to categorize all numeric LGD values into three ordinal categories. As shown below, there are more than 30% of 0 and 1 values.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | df <- read.csv("lgd.csv")  df$lgd <- round(1 - df$Recovery\_rate, 4)  df$lgd\_cat <- cut(df$lgd, breaks = c(-**Inf**, 0, 0.9999, **Inf**), labels = c("L", "M", "H"), ordered\_result = T)  summary(df$lgd\_cat)    #   L    M    H  # 730 1672  143 |

The estimation of a proportional odds model is straightforward with clm() in the ordinal package or polr() in the MASS package. As demonstrated below, in addition to the coefficient for LTV, there are 2 intercepts to differentiate 3 categories.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | m1 <- ordinal::clm(lgd\_cat ~ LTV, data = df)  summary(m1)    #Coefficients:  #    Estimate Std. Error z value Pr(>|z|)  #LTV   2.0777     0.1267    16.4   <2e-16 \*\*\*  #---  #Signif. codes:  0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  #  #Threshold coefficients:  #    Estimate Std. Error z value  #L|M  0.38134    0.08676   4.396  #M|H  4.50145    0.14427  31.201 |

It is important to point out that, in a proportional odds model, it is the cumulative probability that is derived from the linear combination of model variables. For instance, the cumulative probability of LGD belonging to L or M is formulated as  
 **Prob(LGD <= M) = Exp(4.50 – 2.08 \* LTV) / (1 + Exp(4.50 – 2.08 \* LTV))**  
Likewise, we would have  
 **Prob(LGD <= L) = Exp(0.38 – 2.08 \* LTV) / (1 + Exp(0.38 – 2.08 \* LTV))**  
With above cumulative probabilities, then we can calculate the probability of each category as below.  
 **Prob(LGD = L) = Prob(LGD <= L)  
Prob(LGD = M) = Prob(LGD <= M) – Prob(LGD <= L)  
Prob(LGD = H) = 1 – Prob(LGD <= M)**  
The R code is showing the detailed calculation how to convert cumulative probabilities to probabilities of interest.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | cumprob\_L <- exp(df$LTV \* (-m1$beta) + m1$Theta[1]) / (1 + exp(df$LTV \* (-m1$beta) + m1$Theta[1]))  cumprob\_M <- exp(df$LTV \* (-m1$beta) + m1$Theta[2]) / (1 + exp(df$LTV \* (-m1$beta) + m1$Theta[2]))  prob\_L <- cumprob\_L  prob\_M <- cumprob\_M - cumprob\_L  prob\_H <- 1 - cumprob\_M  pred <- data.frame(prob\_L, prob\_M, prob\_H)  apply(pred, 2, mean)    #    prob\_L     prob\_M     prob\_H  #0.28751210 0.65679888 0.05568903 |

After predicting the probability of each category, we would need another sub-model to estimate the conditional LGD for lgd\_cat = “M” with either Beta or Simplex regression.

To better a better illustration of the underlying logic, an example is also provided below, showing how to estimate a Cumulative Logit model by specifying the log likelihood function.

pkgs <- list("maxLik", "VGAM")

sapply(pkgs, require, character.only = T)

df <- read.csv("Downloads/lgd.csv")

df$lgd\_cat <- ifelse(round(1 - df[2], 4) == 0, "L",

ifelse(round(1 - df[2], 4) == 1, "H", "M"))

### DEFINE LOGLIKELIHOOD FUNCTION OF CUMULATIVE LOGIT MODEL ###

# BELOW IS THE SIMPLER EQUIVALENT:

# vglm(sapply(c("L", "M", "H"), function(x) df$lgd\_cat == x) ~ LTV, data = df, family = cumulative(parallel = T))

ll01 <- function(param) {

a1 <- param[1]

a2 <- param[2]

b1 <- param[3]

xb\_L <- a1 - df$LTV \* b1

xb\_M <- a2 - df$LTV \* b1

prob\_L <- exp(xb\_L) / (1 + exp(xb\_L))

prob\_M <- exp(xb\_M) / (1 + exp(xb\_M)) - prob\_L

prob\_H <- 1 - prob\_M - prob\_L

CAT <- data.frame(sapply(c("L", "M", "H"), function(x) assign(x, df$lgd\_cat == x)))

LH <- with(CAT, (prob\_L ^ L) \* (prob\_M ^ M) \* (prob\_H ^ H))

return(sum(log(LH)))

}

Instead of modeling the cumulative probability of each ordered category such that Log(Prob <= Y\_i / (1 – Prob <= Y\_i)) = Alpha\_i – XB, we could also have alternative ways to estimate the categorical probabilities by using Adjacent-Categories Logit and Continuation-Ratio Logit models.

In an Adjacent-Categories Logit model, the functional form can be expressed as Log(Prob = Y\_i / Prob = Y\_j) = Alpha\_i – XB with j = i + 1. The corresponding log likelihood function is given in the code snippet below.

### DEFINE LOGLIKELIHOOD FUNCTION OF ADJACENT-CATEGORIES LOGIT MODEL ###

# BELOW IS THE SIMPLER EQUIVALENT:

# vglm(sapply(c("L", "M", "H"), function(x) df$lgd\_cat == x) ~ LTV, data = df, family = acat(parallel = T, reverse = T))

ll02 <- function(param) {

a1 <- param[1]

a2 <- param[2]

b1 <- param[3]

xb\_L <- a1 - df$LTV \* b1

xb\_M <- a2 - df$LTV \* b1

prob\_H <- 1 / (1 + exp(xb\_M) + exp(xb\_M + xb\_L))

prob\_M <- exp(xb\_M) \* prob\_H

prob\_L <- 1 - prob\_H - prob\_M

CAT <- data.frame(sapply(c("L", "M", "H"), function(x) assign(x, df$lgd\_cat == x)))

LH <- with(CAT, (prob\_L ^ L) \* (prob\_M ^ M) \* (prob\_H ^ H))

return(sum(log(LH)))

}

If we take the probability (Prob = Y\_i) from the Adjacent-Categories Logit and the probability (Prob > Y\_i) from the Cumulative Logit, then we can have the functional form of a Continuation-Ratio Logit model, expressed as Log(Prob = Y\_i / Prob > Y\_i) = Alpha\_i – XB. The log likelihood function is also provided.

ll03 <- function(param) {

a1 <- param[1]

a2 <- param[2]

b1 <- param[3]

xb\_L <- a1 - df$LTV \* b1

xb\_M <- a2 - df$LTV \* b1

prob\_L <- 1 / (1 + exp(-xb\_L))

prob\_M <- 1 / (1 + exp(-xb\_M)) \* (1 - prob\_L)

prob\_H <- 1 - prob\_L - prob\_M

CAT <- data.frame(sapply(c("L", "M", "H"), function(x) assign(x, df$lgd\_cat == x)))

LH <- with(CAT, (prob\_L ^ L) \* (prob\_M ^ M) \* (prob\_H ^ H))

return(sum(log(LH)))

}

After specifying log likelihood functions for aforementioned models, we can use the maxLik::maxLik() function to calculate parameter estimates. It is also shown that, in this particular example, the Cumulative Logit is slightly better than the other alternatives in terms of AIC.

# start = c(a1 = 0.1, a2 = 0.2, b1 = 1.0)

# lapply(list(ll01, ll02, ll03), (function(x) summary(maxLik(x, start = start))))

[[1]]

--------------------------------------------

Estimates:

Estimate Std. error t value Pr(t)

a1 0.38134 0.08578 4.446 8.76e-06 \*\*\*

a2 4.50145 0.14251 31.587 < 2e-16 \*\*\*

b1 2.07768 0.12506 16.613 < 2e-16 \*\*\*

--------------------------------------------

[[2]]

--------------------------------------------

Estimates:

Estimate Std. error t value Pr(t)

a1 0.32611 0.08106 4.023 5.74e-05 \*\*\*

a2 4.05859 0.14827 27.373 < 2e-16 \*\*\*

b1 1.88466 0.11942 15.781 < 2e-16 \*\*\*

--------------------------------------------

[[3]]

--------------------------------------------

Estimates:

Estimate Std. error t value Pr(t)

a1 0.30830 0.08506 3.625 0.000289 \*\*\*

a2 4.14021 0.15024 27.558 < 2e-16 \*\*\*

b1 1.95643 0.12444 15.722 < 2e-16 \*\*\*

--------------------------------------------

# sapply(list(ll01, ll02, ll03), (function(x) AIC(maxLik(x, start = start))))

3764.110 3767.415 3771.373