Hw5

a

```
library(ISLR)
data <- OJ
set.seed(1)
rand <- sample(nrow(data), 800)
train <- data[rand,]
test <- data[-rand,]
```

b

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.2
```

```
svm.model <- svm(Purchase ~ ., data = train, kernel = "linear", cost = 0.01)
summary(svm.model)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "linear", cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  435
##
##  ( 219 216 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

435 observations are used as support vectors (out of the 800 training points). 219 are level CH and 216 are level MM.

c

```
train.pred <- predict(svm.model, train)
train.table <- table(train$Purchase, train.pred)
train.table
```

```
##     train.pred
##       CH  MM
##   CH 420  65
##   MM  75 240
```

```
(train.table[1,2] + train.table[2,1])/sum(train.table)
```

```
## [1] 0.175
```

```
test.pred <- predict(svm.model, test)
test.table <- table(test$Purchase, test.pred)
test.table
```

```
##     test.pred
##       CH  MM
##   CH 153  15
##   MM  33  69
```

```
(test.table[1,2] + test.table[2,1])/sum(test.table)
```

```
## [1] 0.1777778
```

The training error rate is 17.5% and the testing error rate is 17.78%.

d

```
set.seed(1)
svm.tune <- tune(svm, Purchase ~ ., data = train, kernel = "linear", ranges = list(cost = 10^seq(-2, 1,
summary(svm.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  3.162278
##
## - best performance: 0.16875
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.17625 0.02853482
## 2   0.01778279 0.17625 0.03143004
## 3   0.03162278 0.17125 0.02829041
## 4   0.05623413 0.17625 0.02853482
## 5   0.10000000 0.17250 0.03162278
## 6   0.17782794 0.17125 0.02829041
## 7   0.31622777 0.17125 0.02889757
## 8   0.56234133 0.17125 0.02703521
```

```
## 9   1.00000000 0.17500 0.02946278
## 10  1.77827941 0.17375 0.02729087
## 11  3.16227766 0.16875 0.03019037
## 12  5.62341325 0.17375 0.03304563
## 13 10.00000000 0.17375 0.03197764
```

The cost that minimizes error is at cost = 3.16, therefore that is the optimal cost.

e

```
svm.new <- svm(Purchase ~ ., data = train, kernel = "linear", cost = 3.16)

newtrain.pred <- predict(svm.new, train)
newtrain.table <- table(train$Purchase, newtrain.pred)
newtrain.table
```

```
##     newtrain.pred
##       CH  MM
##   CH 423  62
##   MM  70 245
```

```
(newtrain.table[1,2] + newtrain.table[2,1])/sum(newtrain.table)
```

```
## [1] 0.165
```

```
newtest.pred <- predict(svm.new, test)
newtest.table <- table(test$Purchase, newtest.pred)
newtest.table
```

```
##     newtest.pred
##       CH  MM
##   CH 156  12
##   MM  29  73
```

```
(newtest.table[1,2] + newtest.table[2,1])/sum(newtest.table)
```

```
## [1] 0.1518519
```

For best cost (3.16), training error is 16.5% and testing error is 15.2%.

f, with radial

```
svm.modelr <- svm(Purchase ~ ., data = train, kernel = "radial")
summary(svm.modelr)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  373
##
##  ( 188 185 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

379 observations are used as support vectors (out of the 800 training points). 188 are level CH and 185 are level MM.

```
train.predr <- predict(svm.modelr, train)
train.tabler <- table(train$Purchase, train.predr)
train.tabler
```

```
##     train.predr
##       CH  MM
##   CH 441  44
##   MM  77 238
```

```
(train.tabler[1,2] + train.tabler[2,1])/sum(train.tabler)
```

```
## [1] 0.15125
```

```
test.predr <- predict(svm.modelr, test)
test.tabler <- table(test$Purchase, test.predr)
test.tabler
```

```
##     test.predr
##       CH  MM
##   CH 151  17
##   MM  33  69
```

```
(test.tabler[1,2] + test.tabler[2,1])/sum(test.tabler)
```

```
## [1] 0.1851852
```

The training error rate is 15.1% and the testing error rate is 18.5%.

```
set.seed(1)
svm.tuner <- tune(svm, Purchase ~ ., data = train, kernel = "radial", ranges = list(cost = 10^seq(-2, 1
summary(svm.tuner)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##       cost
##  0.5623413
##
## - best performance: 0.16875
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.39375 0.04007372
## 2   0.01778279 0.39375 0.04007372
## 3   0.03162278 0.35750 0.05927806
## 4   0.05623413 0.19500 0.02443813
```

```
## 5   0.10000000 0.18625 0.02853482
## 6   0.17782794 0.18250 0.03291403
## 7   0.31622777 0.17875 0.03230175
## 8   0.56234133 0.16875 0.02651650
## 9   1.00000000 0.17125 0.02128673
## 10  1.77827941 0.17625 0.02079162
## 11  3.16227766 0.17750 0.02266912
## 12  5.62341325 0.18000 0.02220485
## 13 10.00000000 0.18625 0.02853482
```

Choose cost of .562

```
newsvm.modelr <- svm(Purchase ~ ., data = train, kernel = "radial", cost = .562)
summary(newsvm.modelr)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "radial", cost = 0.562)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.562
##
## Number of Support Vectors:  397
##
##  ( 200 197 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
newtrain.predr <- predict(newsvm.modelr, train)
newtrain.tabler <- table(train$Purchase, newtrain.predr)
newtrain.tabler
```

```
##     newtrain.predr
##       CH  MM
##   CH 437  48
##   MM  71 244
```

```
(newtrain.tabler[1,2] + newtrain.tabler[2,1])/sum(newtrain.tabler)
```

```
## [1] 0.14875
```

```
newtest.predr <- predict(newsvm.modelr, test)
newtest.tabler <- table(test$Purchase, newtest.predr)
newtest.tabler
```

```
##     newtest.predr
##       CH  MM
##   CH 150  18
##   MM  30  72
```

```
(newtest.tabler[1,2] + newtest.tabler[2,1])/sum(newtest.tabler)
```

```
## [1] 0.1777778
```

At optimal cost, training error is 14.9% and testing error is 17.8%

g, polynomial w degree = 2

```
svm.modelpoly <- svm(Purchase ~ ., data = train, kernel = "polynomial", degree = 2)
summary(svm.modelpoly)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  447
##
##  ( 225 222 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

447 observations are used as support vectors (out of the 800 training points). 225 are level CH and 222 are level MM.

```
train.predpoly <- predict(svm.modelpoly, train)
train.tablepoly <- table(train$Purchase, train.predpoly)
train.tablepoly
```

```
##     train.predpoly
##       CH  MM
##   CH 449  36
##   MM 110 205
```

```
(train.tablepoly[1,2] + train.tablepoly[2,1])/sum(train.tablepoly)
```

```
## [1] 0.1825
```

```
test.predpoly <- predict(svm.modelpoly, test)
test.tablepoly <- table(test$Purchase, test.predpoly)
test.tablepoly
```

```
##     test.predpoly
##       CH  MM
##   CH 153  15
```

```
##   MM  45  57
```

```
(test.tablepoly[1,2] + test.tablepoly[2,1])/sum(test.tablepoly)
```

```
## [1] 0.2222222
```

The training error rate is 18.25% and the testing error rate is 22.22%.

```
set.seed(1)
svm.tunepoly <- tune(svm, Purchase ~ ., data = train, kernel = "polynomial", degree = 2, ranges = list(
summary(svm.tunepoly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  3.162278
##
## - best performance: 0.1775
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.39125 0.04210189
## 2   0.01778279 0.37125 0.03537988
## 3   0.03162278 0.36500 0.03476109
## 4   0.05623413 0.33750 0.04714045
## 5   0.10000000 0.32125 0.05001736
## 6   0.17782794 0.24500 0.04758034
## 7   0.31622777 0.19875 0.03972562
## 8   0.56234133 0.20500 0.03961621
## 9   1.00000000 0.20250 0.04116363
## 10  1.77827941 0.18500 0.04199868
## 11  3.16227766 0.17750 0.03670453
## 12  5.62341325 0.18375 0.03064696
## 13 10.00000000 0.18125 0.02779513
```

Optimal cost that minimizes error is at 3.16.

```
newsvm.modelpoly <- svm(Purchase ~ ., data = train, kernel = "polynomial", degree = 2, cost = 3.16)
summary(newsvm.modelpoly)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = train, kernel = "polynomial",
##     degree = 2, cost = 3.16)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  3.16
##      degree:  2
##      coef.0:  0
##
```

```
## Number of Support Vectors:  385
##
##  ( 197 188 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
newtrain.predpoly <- predict(newsvm.modelpoly, train)
newtrain.tablepoly <- table(train$Purchase, newtrain.predpoly)
newtrain.tablepoly
```

```
##     newtrain.predpoly
##       CH  MM
##   CH 451  34
##   MM  90 225
```

```
(newtrain.tablepoly[1,2] + newtrain.tablepoly[2,1])/sum(newtrain.tablepoly)
```

```
## [1] 0.155
```

```
newtest.predpoly <- predict(newsvm.modelpoly, test)
newtest.tablepoly <- table(test$Purchase, newtest.predpoly)
newtest.tablepoly
```

```
##     newtest.predpoly
##       CH  MM
##   CH 154  14
##   MM  41  61
```

```
(newtest.tablepoly[1,2] + newtest.tablepoly[2,1])/sum(newtest.tablepoly)
```

```
## [1] 0.2037037
```

At optimal cost, the training error is 15.5% and the testing error is 20.37%.

h

On this data,

Linear: At optimal cost, training error is 16.5% and testing error is 15.2%.

Radial: At optimal cost, training error is 14.9% and testing error is 17.8%

Polynomial: At optimal cost, the training error is 15.5% and the testing error is 20.37%.

It appears that on average, the linear kernel has the lowest testing error on the data. However, I would like to repeat these trials over multiple training samples at different seeds as the low training error for the radial kernel may suggest that it is results in the best results.

STA 325: Homework 5

DUE: 11:59pm, December 10 (on Sakai)
COVERAGE: ISL Chapter 9, Multicategory Modeling

1. **[30 points]** ISL Chapter 9, Question 8.

2. **[20 points]** Let's take a closer look at the baseline-logit model, which extends logistic regression for nominal categorical responses. Suppose we have J categories which are unordered. Let $Y(x) \in \{1, \cdots, J\}$ be the random variable for the categorical response at a single predictor x, and let $\pi_j(x) = \mathbb{P}[Y(x) = j]$, $j = 1, \cdots, J$. Using the first category as a baseline, the baseline-logit model assumes:

$$\log\left(\frac{\pi_j(x)}{\pi_1(x)}\right) = \alpha_j + \beta_j x, \quad j = 2, \cdots, J. \tag{1}$$

(a) **[6 points]** Suppose the slope coefficients $\beta_2 = -0.1$ and $\beta_3 = 0.5$. Interpret these two coefficients for the baseline-logit model. What can you say about the log-odds $\log(\pi_2(x)/\pi_3(x))$ as x changes? (*Hint*: see Equation (6.2) in notes.)

Answer: $\beta_2 = -0.1$: the log-odds of category 2 vs. category 1 changes by -0.1 from a one-unit increase in x; $\beta_3 = 0.5$: the log-odds of category 3 vs. category 1 changes by 0.5 from a one-unit increase in x. From Equation (6.2), we have:

$$\log(\pi_2(x)/\pi_3(x)) = (\alpha_2 - \alpha_3) + (\beta_2 - \beta_3)x,$$

so the log-odds of category 2 vs. category 3 changes by -0.6 from a one-unit increase in x.

(b) **[6 points]** Show that the probabilities from the baseline-logit model (1) follow:

$$\pi_j(x) = \frac{\exp\{\alpha_j + \beta_j x\}}{\sum_{k=1}^{J} \exp\{\alpha_k + \beta_k x\}}, \quad j = 1, \cdots, J,$$

where $\alpha_1 = \beta_1 = 0$. (*Hint*: Solve (1) in terms of $\pi_j(x)$, then sum together the equations for $j = 2, \cdots, J$. Using the fact that $1 - \sum_{j=2}^{J} \pi_j(x) = \pi_1(x)$, solve the resulting equation in terms of $\pi_1(x)$, then use this to get the expression for $\pi_j(x)$.)

Answer: Since:

$$\pi_j(x) = \pi_1(x) \exp\{\alpha_j + \beta_j x\},$$

it follows that:

$$\sum_{k=2}^{J} \pi_k(x) = 1 - \pi_1(x) = \pi_1(x) \sum_{k=2}^{J} \exp\{\alpha_k + \beta_k x\}.$$

Solving for $\pi_1(x)$, we get:

$$\pi_1(x) = \frac{1}{1 + \sum_{k=2}^{J} \exp\{\alpha_k + \beta_k x\}}.$$

Plugging this into (1), we get

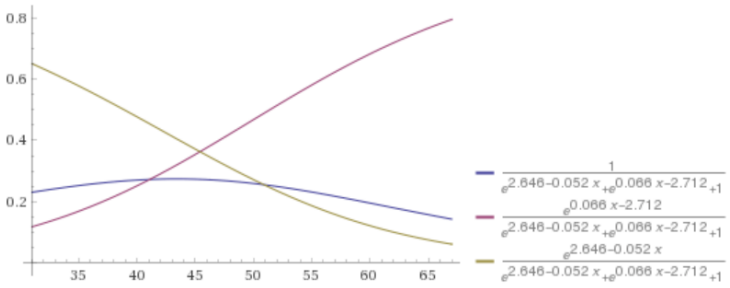
$$\pi_j(x) = \frac{\exp\{\alpha_j + \beta_j x\}}{\sum_{k=1}^{J} \exp\{\alpha_k + \beta_k x\}}, \quad j = 1, \cdots, J.$$

(c) **[8 points]** Suppose we are interested in how a student's score on a writing test is associated with the type of program he / she is in ("General", "Academic" or "Vocational"). The first quantity is measured by the continuous variable `write`, and the second is measured by the nominal categorical variable `prog` with levels 1 (baseline), 2 and 3. R gives the following output on the model fit `model <- multinom(prog ~ write)`:

```
> summary(model)
Call:
multinom(formula = prog ~ write, data = mdata)

Coefficients:
  (Intercept)      write
2   -2.712081  0.0660027
3    2.646492 -0.0517980

Std. Errors:
  (Intercept)      write
2    1.132812 0.02100153
3    1.127455 0.02251449

Residual Deviance: 371.0217
AIC: 379.0217
```

Using this output, plot out the fitted probabilities π_1(`write`), π_2(`write`) and π_3(`write`), as `write` changes from a minimum score of 31 to a maximum score of 67. Interpret your results in terms of the problem.

Answer: Blue is "General"; purple is "Academic"; yellow is "Vocational".



The data suggests that an increase in a student's writing score does not change the probability that he / she is in a "General" program too much. However, an increase in writing score greatly increases the probability that he / she is in an "Academic" program, and greatly decreases the probability that he / she is in a "Vocational" program.

3. **[24 points]** Let's take a closer look at the cumulative-logit model, which extends logistic regression for ordinal categorical responses. Suppose we have J categories which are ordered. Let $Y(x) \in \{1, \cdots, J\}$ be the random variable for the categorical response at a single predictor x, and let $\bar{\pi}_j(x) = \mathbb{P}[Y(x) \leq j]$, $j = 1, \cdots, J$. The proportional-odds cumulative-logit model (as implemented in the R function `polr`) assumes:

$$\text{logit}\{\bar{\pi}_j(x)\} = \alpha_j - \beta x, \quad j = 1, \cdots, J-1. \tag{2}$$

(a) **[6 points]** Give an expression for the cumulative probabilities $\bar{\pi}_j(x)$, $j = 1, \cdots, J-1$, and also for $\bar{\pi}_J(x)$. Using this, give an expression for the class probabilities $\pi_j(x)$, $j = 1, \cdots, J$.

Answer: We can take the inverse-logit transform of (2) to get:

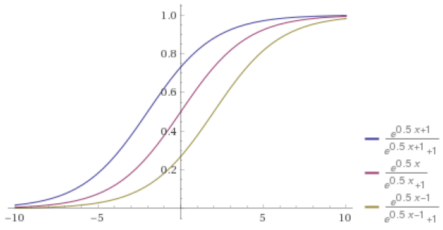
$$\bar{\pi}_j(x) = \frac{\exp\{\alpha_j - \beta x\}}{1 + \exp\{\alpha_j - \beta x\}}, \quad j = 1, \cdots, J-1.$$

Note that $\bar{\pi}_J(x) = \mathbb{P}[Y(x) \leq J] = 1$, so we don't need to model the last cumulative logit. The class probabilities can then be written as:

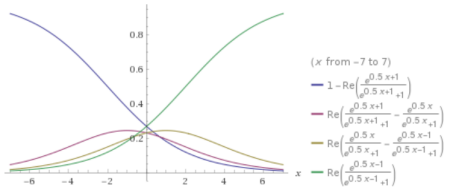
$$\pi_j(x) = \begin{cases} 1 - \frac{\exp\{\alpha_{J-1} - \beta x\}}{1+\exp\{\alpha_{J-1} - \beta x\}}, & \text{if } j = J, \\ \frac{\exp\{\alpha_j - \beta x\}}{1+\exp\{\alpha_j - \beta x\}} - \frac{\exp\{\alpha_{j-1} - \beta x\}}{1+\exp\{\alpha_{j-1} - \beta x\}}, & \text{if } j = 2, \cdots, J-1, \\ \frac{\exp\{\alpha_1 - \beta x\}}{1+\exp\{\alpha_1 - \beta x\}}, & \text{if } j = 1. \end{cases}$$

(b) **[10 points]** Suppose we have $J = 4$ ordered classes, with $\alpha_1 = -1$, $\alpha_2 = 0$, $\alpha_3 = 1$ and $\beta = -0.5$. Plot the cumulative probabilities $\bar{\pi}_j(x)$, $j = 1, \cdots, 3$ and the class probabilities $\pi_j(x)$, $j = 1, \cdots, 4$ as a function of x. Interpret the slope parameter $\beta = -0.5$. How do the class probabilities change as x increases?

Answer: Cumulative probabilities (yellow: $\bar{\pi}_1$; purple: $\bar{\pi}_2$; blue: $\bar{\pi}_3$):



Class probabilities (green: π_1; yellow: π_2; purple: π_3; blue: π_4):



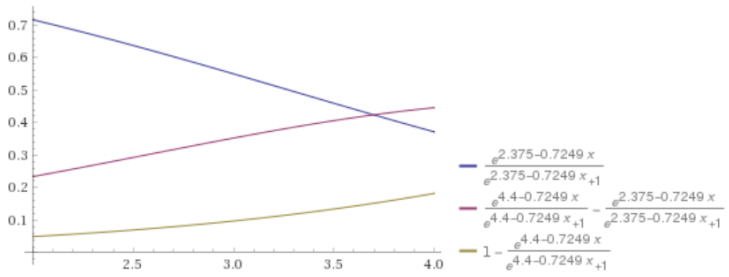
The slope parameter $\beta = -0.5$ represents the decrease in log-odds of falling into or below any category with one-unit increase in x. As x increases, the cumulative logit model places greater probabilities on lower-valued categories.

(c) **[8 points]** Suppose we are interested in how a college junior's GPA influences whether or not he / she applies to graduate school. The first quantity is measured by the continuous variable `gpa`, and the second is measured by the ordinal categorical variable `apply` with levels 1 (unlikely), 2 (somewhat likely) and 3 (very likely). R gives the following output on the model fit `model <- polr(apply ~ gpa)`:

```
> summary(model)
Call:
polr(formula = apply ~ gpa, data = dat, Hess = TRUE)

Coefficients:
     Value Std. Error t value
gpa 0.7249     0.2493   2.908

Intercepts:
                           Value  Std. Error t value
unlikely|somewhat likely   2.3748 0.7570     3.1372
somewhat likely|very likely 4.3998 0.7815     5.6301

Residual Deviance: 732.60
AIC: 738.60
```

Using this output, plot out the fitted probabilities $\pi_1(\texttt{gpa})$, $\pi_2(\texttt{gpa})$ and $\pi_3(\texttt{gpa})$, as `gpa` changes 2.0 to 4.0. Interpret your results in terms of the problem.

Answer: Class probabilities (blue: unlikely; purple: somewhat likely; yellow: very likely):



The data suggests that a student with higher `gpa` is less likely to consider graduate school "unlikely", and more likely to consider graduate school "somewhat likely" or "very likely". In other words, student with higher `gpa` will be more inclined to go to graduate school.

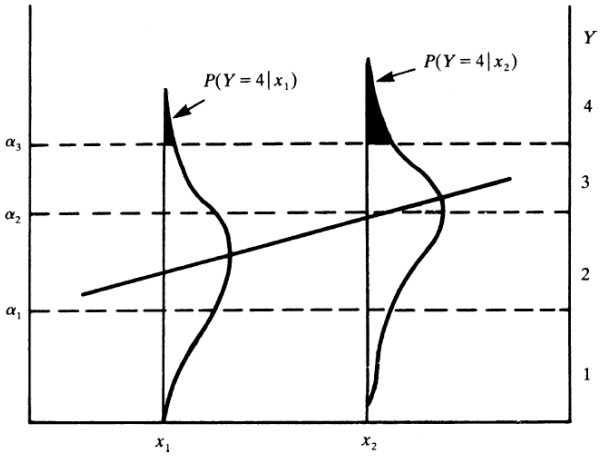
(d) **[BONUS 6 points]** Suppose the underlying generating mechanism for the categorical response $Y(x)$ follows the two-step procedure. First, for a fixed value of x, simulate the latent (i.e., unobserved) random variable $Z(x)$ from the normal distribution:

$$Z(x) = \mathcal{N}(\beta x, 1).$$

Next, let $Y(x)$ be the following discretization of $Z(x)$:

$$Y(x) = \begin{cases} 1, & \text{if } Z(x) \in (-\infty, \alpha_1] \\ 2, & \text{if } Z(x) \in (\alpha_1, \alpha_2] \\ 3, & \text{if } Z(x) \in (\alpha_2, \alpha_3] \\ \vdots & \\ J, & \text{if } Z(x) \in (\alpha_{J-1}, +\infty) \end{cases} .$$

Another way to view this is that $Y(x)$ is *binned* into the predetermined intervals $(-\infty, \alpha_1]$, $(\alpha_1, \alpha_2]$, ... , $(\alpha_{J-1}, +\infty)$; see below:



Under this generative model, show that the cumulative probabilities $\pi_j(x)$ follow:

$$G\{\bar{\pi}_j(x)\} = \alpha_j - \beta x, \quad j = 1, \cdots, J-1,$$

for some function G (specify what G is). Compare and contrast this new model with the model in (2). What needs to be changed in the generating mechanism to yield the cumulative logit model (2)?