Demo

Suyoung Park (spark148)

12/2/2020

Another apporach for finding LCM and linearity.

While I played with the data, I found subsetting unique values based on the decision boundary may produce the same results as glmdr. Below is demo.

Agresti's example (Completely Degenerated)

To my understanding, LCM model is the glm model with all non-problematic points (i.e. linearity is `TRUE`). Thus, to find the LCM model, we need to find the problematic points and technically, `linearity`. In `glmdr` function, we firstly find the MLE in completion using custom Newton method. Then based on that, we approximate the null eigenvector of Fisher information to determine `linearity`.

```
x <- c(10,20,30,40,60,70,80,90)
y <- c(rep(0,4),rep(1,4))
glmdr_model <- glmdr(y~x, family="binomial")
x[!glmdr_model$linearity]
```

```
## [1] 10 20 30 40 60 70 80 90
```

As I discussed last meeting, I think that finding unique values of each distinct problematic predictor (i.e. variable that casues the separation) in terms of response variable are another way to find the problematic points (linearity == FALSE). For example, line 46 is the equivalent to the linearity==FALSE in logistic regression **when complete separation exists**.

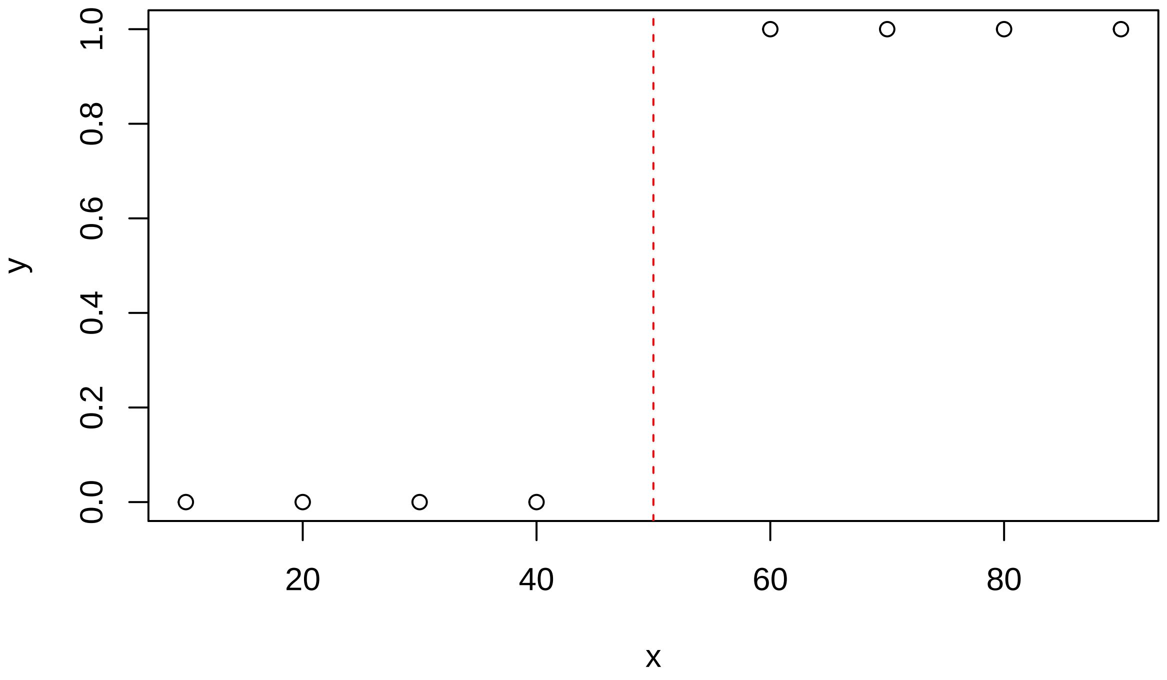
- Comment: when quasi-separation presents or no separation exists, below code does NOT work but I think I can construct generic code using decision boundary which can be obtained by solving $logit^{-1}(\theta) \geq 0.5$ where $\theta = X\beta$ (line 53).

```
dat <- data.frame(cbind(y,x))
cases <- data.frame(unique(x))
names(cases) <- "x"
cases[sapply(1:length(cases),function(i) nlevels(as.factor(
  inner_join(dat,cases[i,,drop=FALSE], by=c("x"))$y))==1),] #line 46
```

```
## [1] 10 20 30 40 60 70 80 90
```

Visually, what I tried to do in line 46 was to see if there are both 0,1 in response variable for each space on decision boundary (i.e., check if there are 0 or 1 in left side of red-dashed line (decision boundary) and right side of red-dashed line).

```
decision_boundary = -coef(glmdr_model$om)[1]/coef(glmdr_model$om)[2] # line 53
plot(y~x)
abline(v=decision_boundary,col="red",lty=2)
```



Maize data

In the Maize data, I can obtain the problematic points using the same code (line 70th). In here, we do not need to find the decision boundary because dimension is already large enough each unique observation in our data plays a role of decision boundary (this is very "unmathematical" statement but I think I can construct formal way using series of decision boundaries). In line 70th, we can see they are the same.

```
dat <- read_xlsx("../manuscript/Supplementary/Corn_data/Combined_Final_Product.xlsx")
names(dat)[c(10,8)] <- c("Kernel.color","Pop.structure")
Xind <- 11:ncol(dat)
foo <- dat[, c(10,8,Xind[-c(15,19,22,23,24)])]
foo$Pop.structure <- factor(foo$Pop.structure)
glmdr_model <- glmdr(Kernel.color~.,data=foo, family="binomial")
cases <- unique(foo[,2:ncol(foo)])
problematic_points <- cases[sapply(1:nrow(cases),function(i) nlevels(as.factor(
  inner_join(foo[,],cases[i,], by=names(cases))$Kernel.color))==1),] #line 70
identical(problematic_points, foo[!glmdr_model$linearity,-1])
```

```
## [1] TRUE
```

After all, I wanted to find the way to express "easier" way of our method in high dimension case as it is quite hard to make the reader (who may not be familiar with complete separation) understand how our approach handles the complete separation (e.g., what is linearity & meaning of the null eigenvector of Fisher information). If we can argue that complete separation happens when there is no both response variable values (0,1) for each space divided by decision boundary (this is reason why I showed the contingency table in the logistc regression in the meeting), I think it is little bit easier to explain our method in complete separation. Furthermore we may connect complete separation to the incomplete tables (zeros in the cells) cases in Poisson regression (i.e., structural zeros is Poisson version of complete separation). But I am not entirely sure about this idea. We can just stick to the Agresit example and covered what we discussed previously. But, I thought it could be possible to provide eaiser explanation if 1 line of code (46 or 70th) can do the similar thing as glmdr.

Fisher information

My idea was I wanted to develop our parer in terms of the Fisher information. For example, diagonal value of Fisher information shows the variance of score function of each predictor and for complete separation case, they are going to be small (or zero).

- I am not entirely sure this is true but based on the Wikipedia (https://en.wikipedia.org/wiki/Fisher_information), "the Fisher information is a way of measuring the amount of information that an observable random variable X carries about an unknown parameter of a distribution that models X" So, I thought in the Agresti example, X values less than 50 does not have any information on response variable 1 and X values larger than 50 does not have any information on response variable 0. Therefore, fisher information of X is 0 meaning X has no ability to estimate parameter (coefficient) and that is reason why MLE does not exist. Also, Wikipedia says "the Fisher information may be seen as the curvature of the support curve. Near the maximum likelihood estimate, low Fisher information therefore indicates that the maximum appears"blunt", that is, the maximum is shallow and there are many nearby values with a similar log-likelihood." So, I think blunt means here flat likelihood function (since it is at infinity). But, I am not sure if I totally misunderstood the concept here.

Agresti example

In this example, the model is completely degenerated and MLE for x (and intercept) does not exist. Therefore, **diagonal** elements of variance-covariance matrix of β goes to infinity, which results in extremely small value in Fisher information.

```
mod <- glmdr(y~x, family="binomial")
coef(summary(mod$om))
```

```
##                Estimate Std. Error       z value  Pr(>|z|)
## (Intercept) -118.15802 296046.187 -0.0003991202 0.9996815
## x              2.36316   5805.939  0.0004070247 0.9996752
```

```
fish <- solve(vcov(mod$om))
fish
```

```
##              (Intercept)            x
## (Intercept) 2.966457e-10 1.483229e-08
## x           1.483229e-08 7.712801e-07
```

Maize data

For Maize data, all coefficients of markers (starting with S6_) seem going to infinity. But their diagonal value of fisher information does not go to zero.

```
mod <- glmdr(Kernel.color ~ ., data=foo,family="binomial")
coef(summary(mod$om))
```

```
##                                 Estimate   Std. Error      z value
## (Intercept)                  -303.446767 2.968351e+04 -0.010222737
## Pop.structurepopcorn            1.464072 1.068327e+00  1.370434021
## Pop.structurestiff stalk        1.218572 6.803999e-01  1.790963831
## Pop.structuresweet corn         1.127600 6.810082e-01  1.655780087
## Pop.structuretropical          -3.246059 3.910769e-01 -8.300306642
## Pop.structureunclassified      -0.669141 3.597358e-01 -1.860090005
## S6_82170011                 -3226.440417 1.177565e+05 -0.027399263
## S6_82170814                   105.107671 6.298940e+03  0.016686565
## S6_82170859               -2985.797822 1.458763e+05 -0.020468013
## S6_82170897                 155.618756 5.719110e+03  0.027210310
## S6_82170900                2427.069642 1.308275e+05  0.018551682
## S6_82170957                 -59.459936 5.803041e+03 -0.010246341
## S6_82171038                 574.071107 4.685339e+04  0.012252498
## S6_82174349                -101.059628 1.666551e+04 -0.006063998
## S6_82174376                 172.496561 1.795233e+04  0.009608591
## S6_82174378                 128.199000 1.691886e+04  0.007577282
## S6_82176123                 166.255019 3.684535e+04  0.004512239
## S6_82185767                1215.429709 5.105783e+04  0.023804962
## S6_82185973                -413.714219 2.005967e+04 -0.020624180
## S6_82186654                 254.753488 2.365849e+04  0.010767951
## S6_82217770                -241.861920 9.601204e+03 -0.025190790
## S6_82217918               -2271.585091 1.040179e+05 -0.021838413
## S6_82218018                 404.487582 7.478302e+04  0.005408816
## S6_82218219                -166.396368 1.329508e+04 -0.012515639
## S6_82243856                4168.701381 1.554828e+05  0.026811335
##                                Pr(>|z|)
## (Intercept)                9.918436e-01
## Pop.structurepopcorn       1.705515e-01
## Pop.structurestiff stalk   7.329910e-02
## Pop.structuresweet corn    9.776636e-02
## Pop.structuretropical      1.038430e-16
## Pop.structureunclassified  6.287279e-02
## S6_82170011                9.781413e-01
## S6_82170814                9.866867e-01
## S6_82170859                9.836700e-01
## S6_82170897                9.782920e-01
## S6_82170900                9.851987e-01
## S6_82170957                9.918247e-01
## S6_82171038                9.902242e-01
## S6_82174349                9.951617e-01
## S6_82174376                9.923336e-01
## S6_82174378                9.939543e-01
## S6_82176123                9.963998e-01
## S6_82185767                9.810082e-01
## S6_82185973                9.835455e-01
## S6_82186654                9.914086e-01
## S6_82217770                9.799028e-01
## S6_82217918                9.825769e-01
## S6_82218018                9.956844e-01
## S6_82218219                9.900142e-01
## S6_82243856                9.786102e-01
```

```
diag(solve(vcov(mod$om)))
```

```
##               (Intercept)      Pop.structurepopcorn
##               164.1241946                 0.9800047
##  Pop.structurestiff stalk   Pop.structuresweet corn
##                 2.9237571                 2.9166956
##     Pop.structuretropical Pop.structureunclassified
##                31.2277055               117.7651455
##               S6_82170011               S6_82170814
##               164.1241939               164.1241940
##               S6_82170859               S6_82170897
##                164.1241943               164.1241918
##                S6_82170900               S6_82170957
##                164.1241944               164.1241938
##                S6_82171038               S6_82174349
##                164.1241943               164.1241941
##                S6_82174376               S6_82174378
##                164.1241942               164.1241941
##                S6_82176123               S6_82185767
##                164.1241943               164.1241941
##                S6_82185973               S6_82186654
##                164.1241938               164.1241940
##                S6_82217770               S6_82217918
##                164.1241921               164.1241944
##                S6_82218018               S6_82218219
##                164.1241942               164.1241939
##                S6_82243856
##                164.1241939
```

This is the same for the model without pop structure.

```
mod <- glmdr(Kernel.color ~ .-Pop.structure, data=foo,family="binomial")
coef(summary(mod$om))
```

```
##                 Estimate Std. Error      z value  Pr(>|z|)
## (Intercept)   -297.26234  13990.828 -0.021246945 0.9830487
## S6_82170011 -3276.22704  81965.440 -0.039970835 0.9681164
## S6_82170814   109.36173   4330.887  0.025251578 0.9798543
## S6_82170859 -2994.78259  91974.308 -0.032561078 0.9740246
## S6_82170897   155.98543   3952.073  0.039469267 0.9685163
## S6_82170900  2452.02908  91331.300  0.026847631 0.9785813
## S6_82170957   -55.55412   4100.588 -0.013547843 0.9891907
## S6_82171038   570.01211  34548.335  0.016498975 0.9868363
## S6_82174349   -94.16950   8065.537 -0.011675540 0.9906845
## S6_82174376   171.04704  10416.368  0.016420987 0.9868985
## S6_82174378   109.23213  12718.655  0.008588340 0.9931476
## S6_82176123   149.61471  27597.366  0.005421340 0.9956744
## S6_82185767  1226.56410  34011.530  0.036063185 0.9712320
## S6_82185973  -421.68381  14068.792 -0.029972993 0.9760886
## S6_82186654   279.37017  19911.669  0.014030475 0.9888057
## S6_82217770  -244.59678   6387.844 -0.038290973 0.9694557
## S6_82217918 -2297.41961  65164.020 -0.035255953 0.9718756
## S6_82218018   402.45315  51598.286  0.007799739 0.9937768
## S6_82218219  -166.47048   9403.911 -0.017702261 0.9858764
## S6_82243856  4224.05559 105385.487  0.040081948 0.9680278
```

```
diag(solve(vcov(mod$om)))
```

```
## (Intercept) S6_82170011 S6_82170814 S6_82170859 S6_82170897 S6_82170900
##    211.5576    211.5576    211.5576    211.5576    211.5576    211.5576
## S6_82170957 S6_82171038 S6_82174349 S6_82174376 S6_82174378 S6_82176123
##    211.5576    211.5576    211.5576    211.5576    211.5576    211.5576
## S6_82185767 S6_82185973 S6_82186654 S6_82217770 S6_82217918 S6_82218018
##    211.5576    211.5576    211.5576    211.5576    211.5576    211.5576
## S6_82218219 S6_82243856
##    211.5576    211.5576
```

So, I wonder this is because scale problem.