ISE 789/OR 791 HW5

Due: 12/6/2019 11:59 PM

1. **Consider the PCB experiment discussed in the class.** *(X.csv contains the independent variables; Y contains the response variable, i.e., the number of bad quality boards among all 400 boards)*
    a. Treat variable A and B as regular predictors and fit a logit GLM model. *(1) Code the IRWLS algorithm. (2) Report the regression coefficients you estimated and submit your code.*
    b. Treat variable A and B as categorical predictors. Use dummy variable technique and fit a logit GLM model.
    (1) Code the IRWLS algorithm. (2) Report the regression coefficients you estimated and submit your code.
2. **PCA.** Given a matrix $X \in R^{20\times10}$, where the sample size is 20 and the dimensionality of each sample is 10 (Please see dataset "Question2.csv").
    a. Apply PCA on X using SVD (i.e., you cannot use function ***pca*** directly, but you may use the function for ***svd***). *Report the result and submit your code.*
    b. Apply PCA on X using Eigen decomposition (i.e., you cannot use functiona ***pca*** directly, but you may use the function for eigen decomposition). *Report the result and submit your code.*

ISE-789 HW-5

Code

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Hide

```
plot(cars)
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

(X.csv contains the independent variables; Y contains the response variable, i.e., the number of bad quality boards among all 400 boards) a. Treat variable A and B as regular predictors and fit a logit GLM model. (1) Code the IRWLS algorithm. (2) Report the regression coefficients you estimated and submit your code. b. Treat variable A and B as categorical predictors. Use dummy variable technique and fit a logit GLM model. (1) Code the IRWLS algorithm. (2) Report the regression coefficients you estimated and submit your code

1.

Hide

```
X = read.csv("C:/Users/deepa/Downloads/X.csv", header = TRUE)
```

Hide

```
head(X)
```

	A <int>	B <int>	C <int>	D <int>	E <int>	F <int>	G <int>	H <int>	Y <int>
1	1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2	5

	A <int>	B <int>	C <int>	D <int>	E <int>	F <int>	G <int>	H <int>	Y <int>
3	1	1	3	3	3	3	3	3	21
4	1	2	1	1	2	2	3	3	9
5	1	2	2	2	3	3	1	1	25
6	1	2	3	3	1	1	2	2	8

6 rows

As Y is continuous, we need to make it proportional.

Hide

```
X$Y=X$Y/400
```

Hide

```
head(X)
```

	A <int>	B <int>	C <int>	D <int>	E <int>	F <int>	G <int>	H <int>	Y <dbl>
1	1	1	1	1	1	1	1	1	0.0025
2	1	1	2	2	2	2	2	2	0.0125
3	1	1	3	3	3	3	3	3	0.0525
4	1	2	1	1	2	2	3	3	0.0225
5	1	2	2	2	3	3	1	1	0.0625
6	1	2	3	3	1	1	2	2	0.0200

6 rows

a.Code the IRWLS algorithm. (2) Report the regression coefficients you estimated and submit your code.

Hide

```
logitMod <- glm( Y~ A+B+C+D+E+F+G+H, data=X, family=binomial(link="logit"))
```

```
non-integer #successes in a binomial glm!
```

Hide

```
summary(logitMod)
```

```

Call:
glm(formula = Y ~ A + B + C + D + E + F + G + H, family = binomial(link = "logit"),
    data = X)

Deviance Residuals:
      Min         1Q     Median         3Q        Max
-0.085083  -0.039915   0.009859   0.030370   0.085055

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.379822  12.448162  -0.513    0.608
A            0.898201   2.716498   0.331    0.741
B           -0.122307   1.607620  -0.076    0.939
C            0.022185   1.577677   0.014    0.989
D            0.165575   1.643271   0.101    0.920
E            0.006002   1.704584   0.004    0.997
F            0.915123   1.706657   0.536    0.592
G           -0.220299   1.654913  -0.133    0.894
H            0.058702   1.661520   0.035    0.972

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.573069  on 17  degrees of freedom
Residual deviance: 0.050917  on  9  degrees of freedom
AIC: 19.717

Number of Fisher Scoring iterations: 7
```

Now let's use IRWLS algorithm

Hide

```
irls <- function(X, y, tol=1e-12, iter=100) {
  # intialize
  int = log(mean(y)/(1-mean(y)))   # intercept
  beta = c(int, rep(0, ncol(X)-1))
  currtol=1
  it = 0
  ll = 0

  while (currtol > tol && it < iter) {
    it = it + 1
    ll_old = ll

    eta = X %*% beta
    mu = plogis(eta)[,1]
    s = mu*(1-mu)
    S = diag(s)
    z = eta + (y-mu)/s
    beta = solve(t(X) %*% S %*% X) %*% (t(X) %*% (S %*% z))

    ll = sum(dbinom(y, prob=plogis(X %*% beta), size=1, log=T))
    currtol = abs(ll-ll_old)
  }

  list(beta=beta, iter=it, tol=currtol, loglik=ll,
       weights=plogis(X %*% beta)*(1-plogis(X %*% beta)))
}
```

Let's see the estimation

Hide

```
x_1=model.matrix(logitMod)
y_1=logitMod$y
```

Hide

```
irls_result = irls(X=x_1, y=y_1, tol=1e-8) # tol set to 1e-8 as in glm default
```

```
non-integer x = 0.002500non-integer x = 0.012500non-integer x = 0.052500non-integer x = 0.022500non-integer x = 0.062500non-integer x = 0.020000non-integer x = 0.050000non-integer x = 0.012500non-integer x = 0.017500non-integer x = 0.060000non-integer x = 0.095000non-integer x = 0.045000non-integer x = 0.020000non-integer x = 0.065000non-integer x = 0.107500non-integer x = 0.150000non-integer x = 0.007500non-integer x = 0.022500non-integer x = 0.002500non-integer x = 0.012500non-integer x = 0.052500non-integer x = 0.022500non-integer x = 0.062500non-integer x = 0.020000non-integer x = 0.050000non-integer x = 0.012500non-integer x = 0.017500non-integer x = 0.060000non-integer x = 0.095000non-integer x = 0.045000non-integer x = 0.020000non-integer x = 0.065000non-integer x = 0.107500non-integer x = 0.150000non-integer x = 0.007500non-integer x = 0.022500Error in while (currtol > tol && it < iter) { :
  missing value where TRUE/FALSE needed
```

b.

glm model

Hide

```
logitMod_dummy <- glm( Y~ factor(A)+factor(B)+C+D+E+F+G+H, data=dummy_variable, family=binomial(link="logit"))
```

```
non-integer #successes in a binomial glm!
```

Hide

```
summary(logitMod_dummy)
```

```
Call:
glm(formula = Y ~ factor(A) + factor(B) + C + D + E + F + G + 
    H, family = binomial(link = "logit"), data = dummy_variable)

Deviance Residuals:
      Min         1Q     Median         3Q        Max  
-0.076527  -0.053105   0.001152   0.026483   0.091238  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.75676   10.92309  -0.527    0.598
factor(A)2   0.90654    2.72795   0.332    0.740
factor(B)2   0.03593    3.18083   0.011    0.991
factor(B)3  -0.23816    3.28060  -0.073    0.942
C           -0.00301    1.62987  -0.002    0.999
D            0.17916    1.66867   0.107    0.914
E            0.01166    1.70958   0.007    0.995
F            0.91885    1.71392   0.536    0.592
G           -0.18714    1.75441  -0.107    0.915
H            0.07298    1.69141   0.043    0.966

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.573069  on 17  degrees of freedom
Residual deviance: 0.047596  on  8  degrees of freedom
AIC: 21.717

Number of Fisher Scoring iterations: 7
```

Hide

```
x_2=model.matrix(logitMod_dummy)
y_2=logitMod_dummy$y
```

IRWLS result

Hide

```
irls_result = irls(X=x_2, y=y_2, tol=1e-8) # tol set to 1e-8 as in glm default
```

```
non-integer x = 0.002500non-integer x = 0.012500non-integer x = 0.052500non-integer x = 0.022500non-integer x = 0.062500non-integer x = 0.020000non-integer x = 0.050000non-integer x = 0.012500non-integer x = 0.017500non-integer x = 0.060000non-integer x = 0.095000non-integer x = 0.045000non-integer x = 0.020000non-integer x = 0.065000non-integer x = 0.107500non-integer x = 0.150000non-integer x = 0.007500non-integer x = 0.022500non-integer x = 0.002500non-integer x = 0.012500non-integer x = 0.052500non-integer x = 0.022500non-integer x = 0.062500non-integer x = 0.020000non-integer x = 0.050000non-integer x = 0.012500non-integer x = 0.017500non-integer x = 0.060000non-integer x = 0.095000non-integer x = 0.045000non-integer x = 0.020000non-integer x = 0.065000non-integer x = 0.107500non-integer x = 0.150000non-integer x = 0.007500non-integer x = 0.022500Error in while (currtol > tol && it < iter) { :
  missing value where TRUE/FALSE needed
```

2.2. PCA. Given a matrix ???? ??? ???? 20×10 , where the sample size is 20 and the dimensionality of each sample is 10 (Please see dataset “Question2.csv”). a. Apply PCA on X using SVD (i.e., you cannot use function pca directly, but you may use the function for svd). Report the result and submit your code. b. Apply PCA on X using Eigen decomposition (i.e., you cannot use functiona pca directly, but you may use the function for eigen decomposition). Report the result and submit your code.

Hide

```
pca_data=read.csv("C:/Users/deepa/Downloads/Question2.csv", header = TRUE)
```

Hide

```
head(pca_data)
```

	X0.0065935 <dbl>	X.0.0026889 <dbl>	X.0.0077365 <dbl>	X0.0015186 <dbl>	X0.0067996 <dbl>	X0.0010038 <dbl>	X.0.00056941 <dbl>	X.0.00009 <dbl>
1	-0.00499670	0.0034703	0.0035010	0.0038276	-0.0032704	-0.0036855	-0.0054957	0.0045416
2	-0.00014448	0.0064473	-0.0098704	0.0032442	-0.0033912	0.0025792	0.0039826	0.0064151
3	0.00102400	0.0039428	-0.0074468	0.0055513	0.0019128	0.0013839	0.0030184	-0.0019474
4	-0.00108080	0.0042566	-0.0039465	-0.0044584	0.0026693	0.0019925	0.0014349	-0.0019652
5	0.00122990	0.0024349	0.0063235	-0.0037689	0.0077163	0.0059569	-0.0064270	-0.0057208
6	0.00128670	-0.0041474	0.0065170	-0.0022826	0.0016390	0.0025624	0.0047508	0.0020233

6 rows | 1-9 of 10 columns

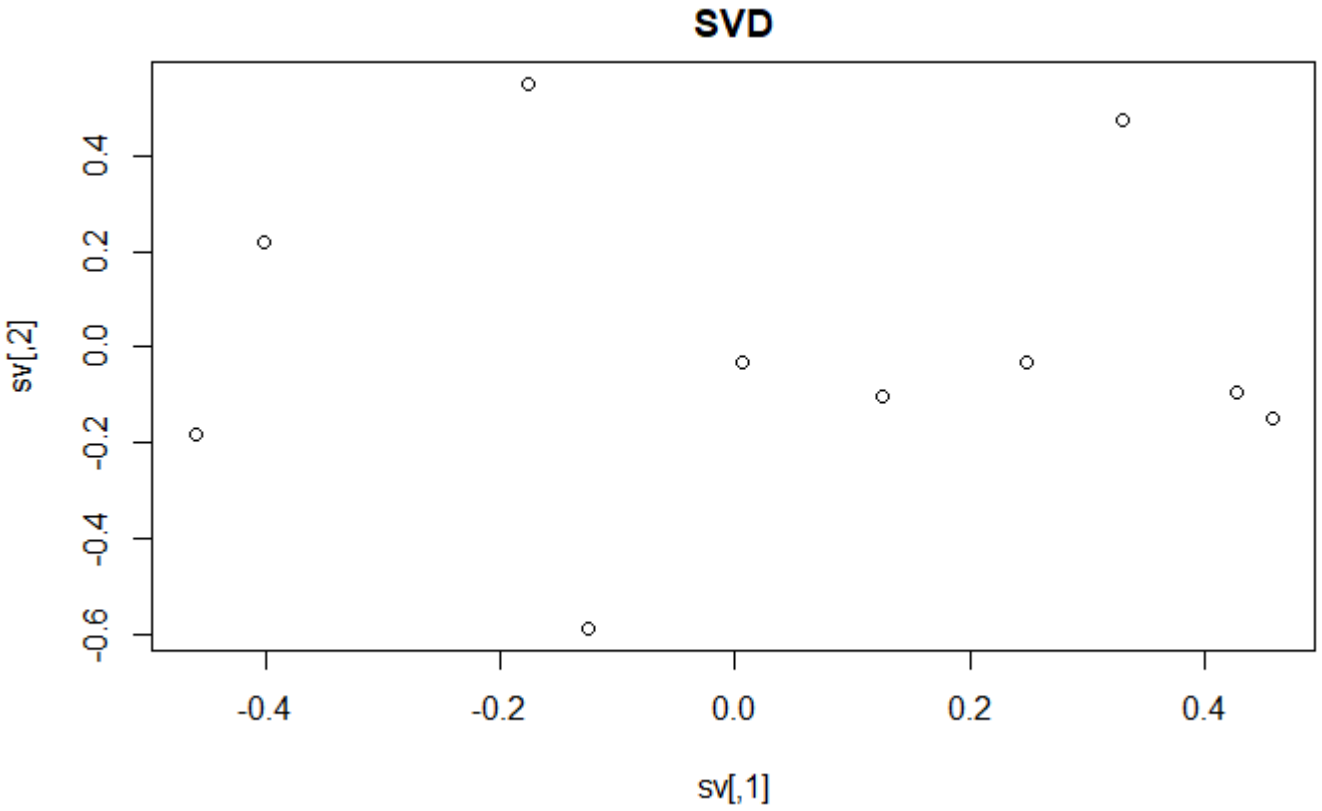
Hide

```
sv=svd(cor(pca_data))$u
sv
```

```
             [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]          [,8]
 [1,] -0.126295664 -0.58800853  0.27740948  0.04534712 -0.04743321  0.31764941 -0.38685099  0.1580265120
 [2,]  0.247771098 -0.03074180  0.45103843 -0.46228974 -0.18857937 -0.48513436  0.20690344  0.2086614143
 [3,] -0.401603090  0.21795677 -0.37995103 -0.19115161  0.16903427 -0.36504082 -0.13507051  0.4830458889
 [4,]  0.427750325 -0.09442560 -0.27425491  0.19992745  0.27772744 -0.45379818 -0.31988672 -0.4419483514
 [5,]  0.005829125 -0.03245529  0.40767183 -0.26160662  0.81130789  0.02147889 -0.07793079 -0.0237804107
 [6,] -0.177152544  0.55095443  0.27154007 -0.25310735 -0.18095288  0.16403329 -0.15036112 -0.5170430920
 [7,]  0.329946845  0.47349241 -0.05090143  0.12384451  0.26368779  0.42071074 -0.07868226  0.3665488276
 [8,]  0.458175923 -0.15115435 -0.25210379 -0.27302534  0.02592745  0.26697801  0.50809873 -0.0000413558
 [9,]  0.125784788 -0.10481890 -0.36698725 -0.67268071 -0.12459677  0.19417037 -0.47065224 -0.0455132769
[10,] -0.460841179 -0.18505603 -0.24345949 -0.18122063  0.28247215  0.10588080  0.41522921 -0.3138595061
              [,9]       [,10]
 [1,] -0.4368702764  0.30108177
 [2,] -0.3592576643 -0.18489118
 [3,] -0.1369140263  0.42208752
 [4,] -0.3212532094  0.09493117
 [5,]  0.3074691749  0.06648670
 [6,] -0.2055346165  0.36801016
 [7,] -0.4458319707 -0.25133192
 [8,]  0.0007920282  0.54682135
 [9,]  0.1561634058 -0.29122794
[10,] -0.4451894679 -0.31803809
```

Hide

```
plot(sv, main = "SVD")
```



B. Eigen vector

Hide

```
eigen(cor(pca_data))$vectors
```

```
             [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]          [,8]
 [1,] -0.126295664  0.58800853 -0.27740948  0.04534712  0.04743321  0.31764941 -0.38685099 -0.1580265120
 [2,]  0.247771098  0.03074180 -0.45103843 -0.46228974  0.18857937 -0.48513436  0.20690344 -0.2086614143
 [3,] -0.401603090 -0.21795677  0.37995103 -0.19115161 -0.16903427 -0.36504082 -0.13507051 -0.4830458889
 [4,]  0.427750325  0.09442560  0.27425491  0.19992745 -0.27772744 -0.45379818 -0.31988672  0.4419483514
 [5,]  0.005829125  0.03245529 -0.40767183 -0.26160662 -0.81130789  0.02147889 -0.07793079  0.0237804107
 [6,] -0.177152544 -0.55095443 -0.27154007 -0.25310735  0.18095288  0.16403329 -0.15036112  0.5170430920
 [7,]  0.329946845 -0.47349241  0.05090143  0.12384451 -0.26368779  0.42071074 -0.07868226 -0.3665488276
 [8,]  0.458175923  0.15115435  0.25210379 -0.27302534 -0.02592745  0.26697801  0.50809873  0.0000413558
 [9,]  0.125784788  0.10481890  0.36698725 -0.67268071  0.12459677  0.19417037 -0.47065224  0.0455132769
[10,] -0.460841179  0.18505603  0.24345949 -0.18122063 -0.28247215  0.10588080  0.41522921  0.3138595061
              [,9]       [,10]
 [1,]  0.4368702764  0.30108177
 [2,]  0.3592576643 -0.18489118
 [3,]  0.1369140263  0.42208752
 [4,]  0.3212532094  0.09493117
 [5,] -0.3074691749  0.06648670
 [6,]  0.2055346165  0.36801016
 [7,]  0.4458319707 -0.25133192
 [8,] -0.0007920282  0.54682135
 [9,] -0.1561634058 -0.29122794
[10,]  0.4451894679 -0.31803809
```