CMAP HW2

Wai Laam (Josie) Lui

11/8/2019

Loading Packages

```
library(tidyverse)
library(ROCR) # ROC
library(MASS)
library(wnominate) # for algorithm
library(pscl) # for "readKH()" function
```

Part 1

Load Data

```
scous_conf <- haven::read_dta("PSET 2 Files/conf06.dta")
conf06 <- subset(scous_conf, scous_conf$nominee!="ALITO")
vars <- c("vote", "nominee", "sameprty", "qual", "lackqual", "EuclDist2", "strngprs") # vector of vars
conf <- conf06[vars] # retain only key vars from above object
# Haven version of read_dta seems to resolve the 1-2 scale issue automatically
```

(1) Data Split

```
## From stack exchange
## 80% of the sample size
smp_size <- floor(0.8 * nrow(conf))

## set the seed to make your partition reproducible
set.seed(42578)
train_ind <- sample(seq_len(nrow(conf)), size = smp_size)
train <- conf[train_ind, ]
test <- conf[-train_ind, ]
```

(2) Build Logit Classifier

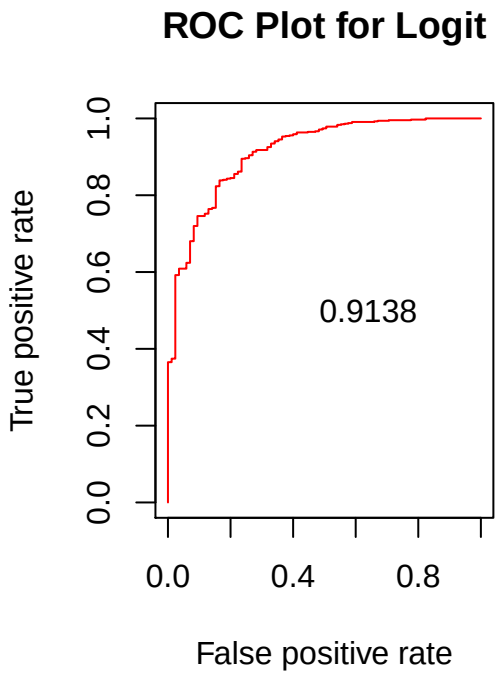
```
# logit on training set
logit <- glm(vote ~ sameprty+strngprs+EuclDist2+qual,
             data = train,
             family = binomial)

# Predictions on test set
logit.probs <- predict(logit, newdata=test,type = "response")
logit.pred <- ifelse(logit.probs > 0.5, 1 ,0)

# confusion matrix for test set at default threshold 0.5
table(logit.pred, test$vote)
```

```
##
## logit.pred   0   1
##          0  39  11
##          1  46 646
```

```
mean(logit.pred == test$vote) # 92% accurate
```

```
## [1] 0.9231806
```

```
# ROC plot
pred <- prediction(logit.probs, as.vector(test$vote))
perf <- performance(pred,"tpr","fpr")
auc <- unlist(slot(performance(pred,"auc"), "y.values"))
plot(perf,main="ROC Plot for Logit",col="red")
legend(0.3,0.6,round(auc,4),
       border="white",cex=1,box.col = "white")
```



At the default threshold 0.5, the logit classifier achieves an accuracy of 92.3 percent. The AOC is 0.9138, which is very close to 1. It means that the two classes (Yes Vote and No Vote) can be well separated based on knowing the covariates.

(3) Build LDA Classifier

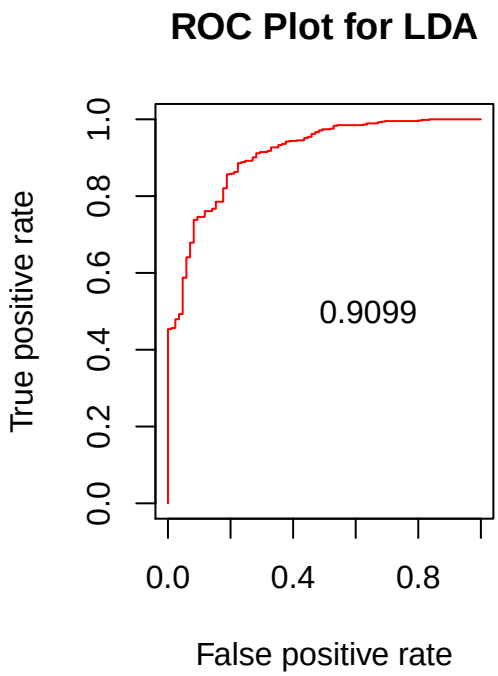
```
# LDA on training set
lda <- lda(vote ~ sameprty+strngprs+EuclDist2+qual,
           data=train)

lda.pred <- predict(lda, newdata=test)

# confusion matrix
table(lda.pred$class, test$vote)
```

```
##
##       0   1
##   0  42  17
##   1  43 640
```

```
# check the classification rate
mean(lda.pred$class == test$vote) # 92% accurate
```

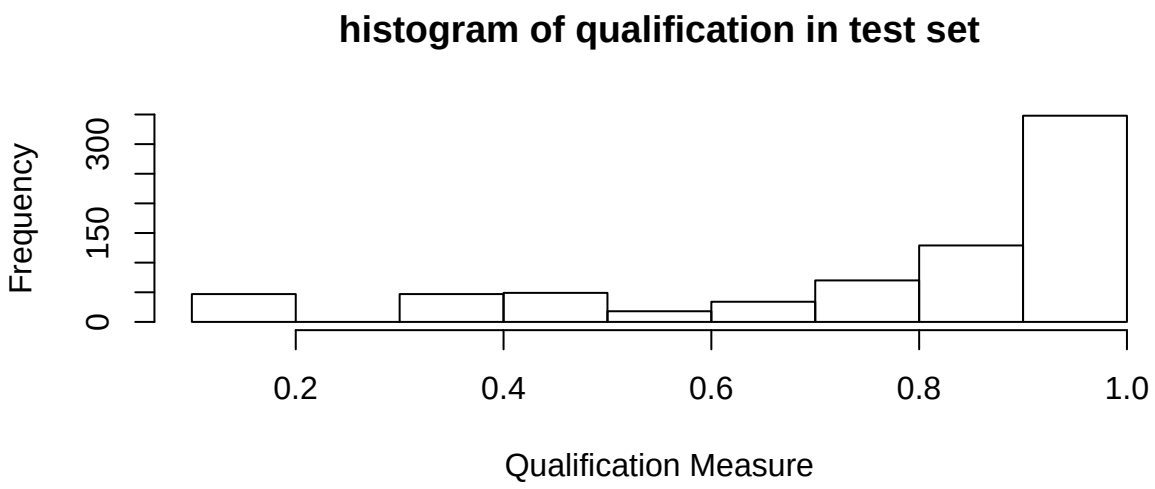
```
## [1] 0.9191375
```

```
pred <- prediction(lda.pred$posterior[,2], as.vector(test$vote))
perf <- performance(pred,"tpr","fpr")
auc <- unlist(slot(performance(pred,"auc"), "y.values"))

plot(perf,main="ROC Plot for LDA",col="red")
legend(0.3,0.6,round(auc,4),border="white",cex=1,box.col = "white")
```



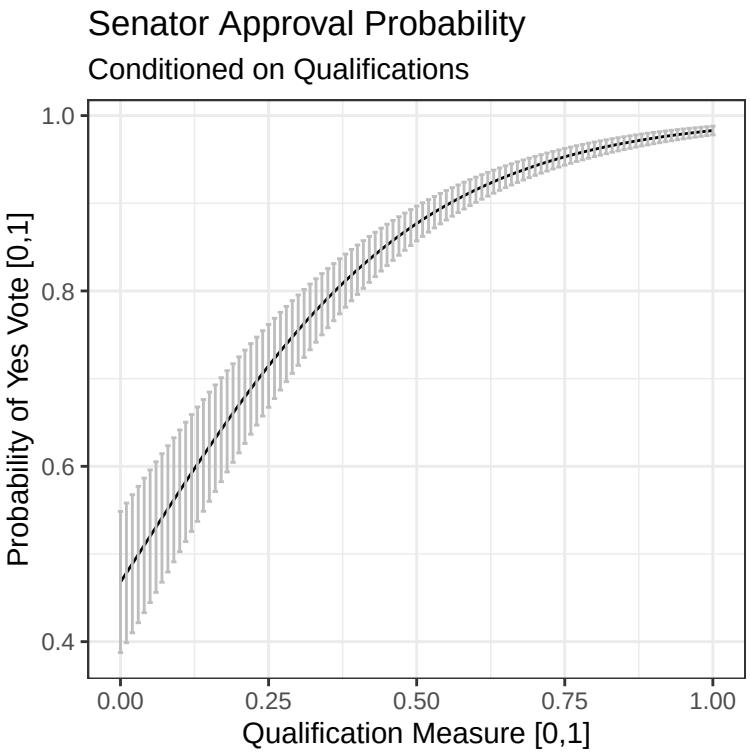
At the default threshold 0.5 (class 2 posterior > class 1 posterior), the LDA classifier achieves an accuracy of 91.9 percent. The AOC is 0.9099, which is very close to 1. It means that the two classes (Yes Vote and No Vote) can be well separated based on knowing the covariates. However, compared to the logit model, the LDA performance is slightly worse. Perhaps the reason is that covariates don't meet the normality assumption that well. For instance - most candidates are quite qualified and the distribution of `qual` is skews heavily to the left.

```
hist(test$qual,main="histogram of qualification in test set",
     xlab="Qualification Measure")
```



(4) Impact of Qualification

```
qual_range <- with(conf,tibble(qual = seq(0,1,by=0.01),
                               sameprty = rep(mean(sameprty),101),
                               EuclDist2 = rep(mean(EuclDist2),101),
                               strngprs = rep(mean(strngprs),101)))
cond.probs <- predict(logit, newdata=qual_range,
                      type = "response", se = TRUE)
qual_df <- cbind(qual_range,cond.probs)
ggplot(qual_df,aes(x=qual,y=fit)) +
  geom_line()+
  labs(title = "Senator Approval Probability",
       subtitle = "Conditioned on Qualifications",
       x = "Qualification Measure [0,1]",
       y = "Probability of Yes Vote [0,1]")+
  geom_errorbar(aes(ymin = fit-1.96*se.fit, ymax = fit+1.96*se.fit),
                color="gray")+
  theme_bw()
```



Based on the logit predictor, holding all other covariates at the mean of the population (whole dataset), probability of a senator voting Yes goes beyond 0.5 even when the qualification score is smaller than 0.1. For a relatively well-qualified nominee whose score is greater than 0.75, his chances of receiving a Yes vote from the "averaage senator" becomes greater than 0.95.

(5) Discussions

Since the logit classifier performs slightly better than LDA, in addition to LDA assumptions not being well-met, we will base the following discussion of results primarily on the logit model.

term	estimate	std.error	statistic	p.value
(Intercept)	-0.9297753	0.2079484	-4.471183	7.8e-06
sameprty	1.3114738	0.1687983	7.769474	0.0e+00
strngprs	1.4393191	0.1509415	9.535610	0.0e+00
EuclDist2	-4.1097010	0.3180060	-12.923343	0.0e+00
qual	4.1832385	0.2546104	16.429957	0.0e+00

First of all, all estimates are on order 1, meaning that all covaraites (which are in range[0,1]) have potential influences comparable in order to one another.

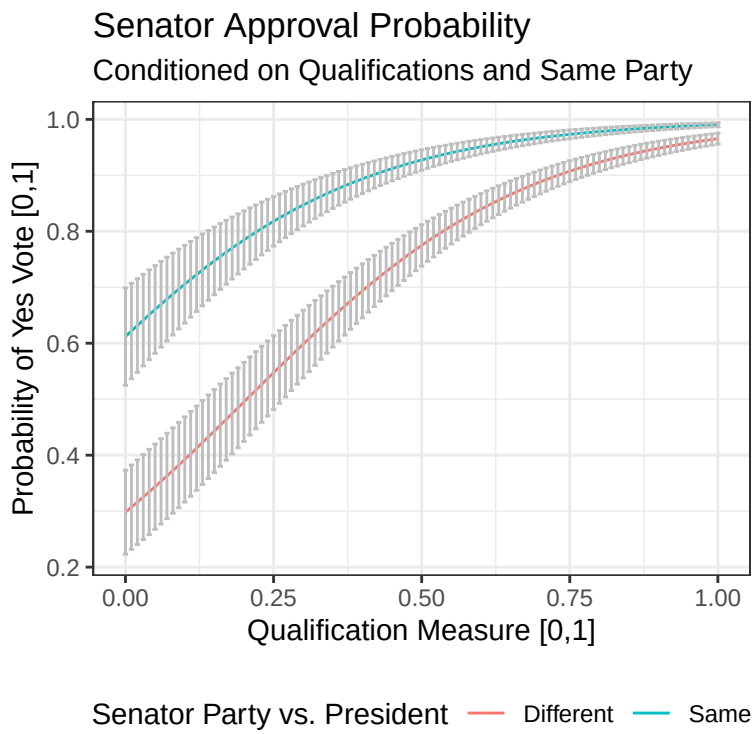
Secondly, we see that the most influential variables (absolute value of the estimates) are ideological distance and qualification. For a completley qualified candidate (`qual` ≈ 1) who is distant from a senator on ideology (`EuclDist2` ≈ 1), the effects of both can almost cancel out. If the ideological distance is big, chances are that `sameprty`=0, since it is the president who nominates the court candidate. In addition, both being on the same party of the president and having a strong president can effectively offset the negative intercept.

From this we can infer that whie ignoring political factors, qualification is the primary influencer on the vote; yet political factors can over-power the influence of qualifications substantially.

Above all, we can see the influence of party politics most clearly from the plot below: while we may think that senators should vote on judge nominations based on merit and merit alone, party affiliation can increase the likelihood of a Yes vote by more than 100% for poorly qualified candidates. The vote is partisan indeed.

(6) Conditioning Qualification on Party

```
qual_range_0 <- with(conf,tibble(qual = seq(0,1,by=0.01),
                                 sameprty = rep(0,101),
                                 EuclDist2 = rep(mean(EuclDist2),101),
                                 strngprs = rep(mean(strngprs),101),
                                 party = "Different"))
qual_range_1 <- qual_range_0 %>% mutate(sameprty = 1,party="Same")
qual_range = bind_rows(qual_range_0,qual_range_1)
cond.probs <- predict(logit, newdata=qual_range,
                      type = "response", se = TRUE)
qual_df <- cbind(qual_range,cond.probs)
ggplot(qual_df,aes(x=qual,y=fit,color=party)) +
  geom_line()+
  labs(title = "Senator Approval Probability",
       subtitle = "Conditioned on Qualifications and Same Party",
       x = "Qualification Measure [0,1]",
       y = "Probability of Yes Vote [0,1]",
       color = "Senator Party vs. President")+
  geom_errorbar(aes(ymin = fit-1.96*se.fit, ymax = fit+1.96*se.fit),
                color="gray")+
  theme_bw()+
  theme(legend.position="bottom")
```



Part 2

W-Nominate Fitting

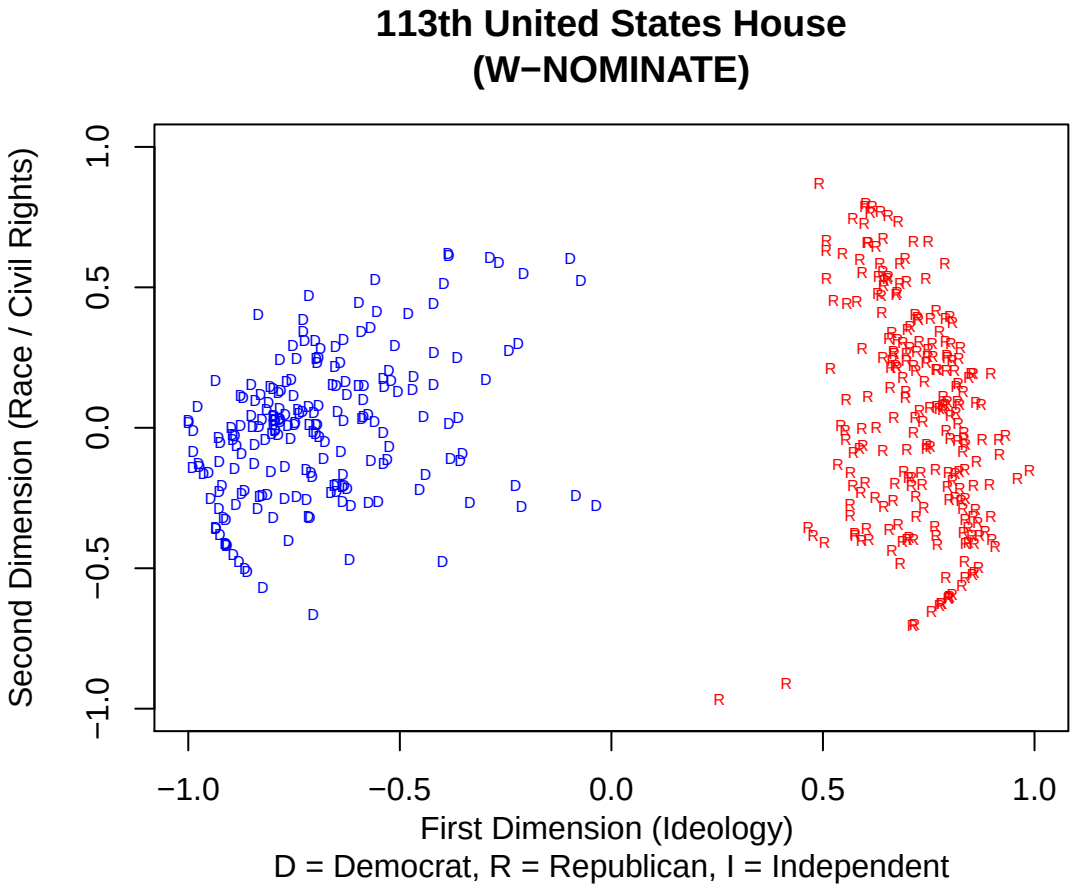
```
house113 <- readKH(
  "PSET 2 Files/hou113kh.ord", # locate the .ord file saved locally dtl=NULL,
  yea=c(1,2,3),
  nay=c(4,5,6),
  missing=c(7,8,9),
  notInLegis=0,
  desc="113th_House_Roll_Call_Data",
  debug=FALSE
)
```

```
## Attempting to read file in Keith Poole/Howard Rosenthal (KH) format.
## Attempting to create roll call object
## 113th_House_Roll_Call_Data
## 445 legislators and 1202 roll calls
## Frequency counts for vote types:
## rollCallMatrix
##      0      1      6      7      9
##  14576 295753 202943    290  21328
```

```
# Fit the algorithm
# run once and stored the results.
# wnom_113 <- wnominate(house113,
#                           dims = 2,
#                           minvotes = 20,
#                           lop = 0.025,
#                           polarity = c(2,2))
# write_rds(wnom_113,"wnom_113_opt.rds")
wnom_113 = read_rds("wnom_113_opt.rds")
```

Now we are ready to plot the members of the house.

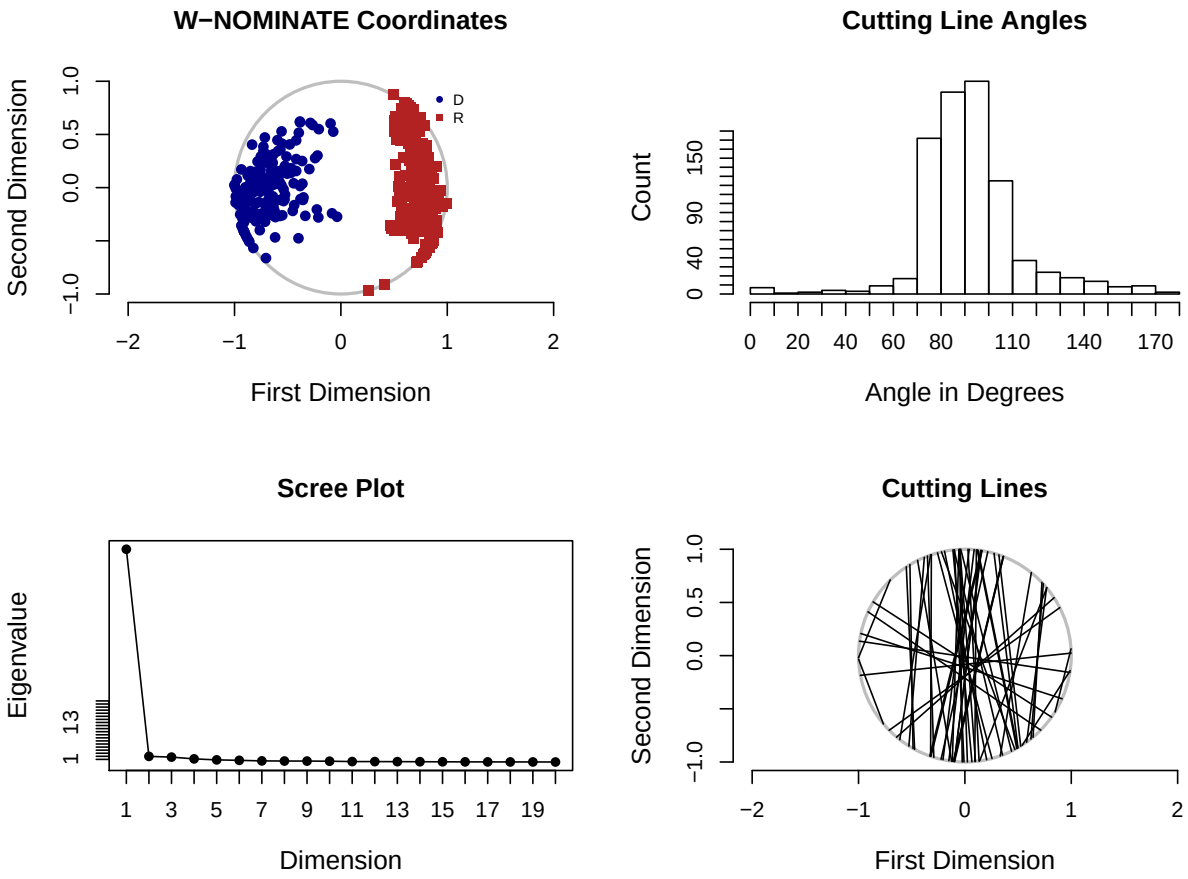
```
# Plot
# store a few things for plotting
wnom1 <- wnom_113$legislators$coord1D
wnom2 <- wnom_113$legislators$coord2D
party <- house113$legis.data$party

# custom plot
plot(wnom1, wnom2,
     main="113th United States House\n(W-NOMINATE)",
     xlab="First Dimension (Ideology) \nD = Democrat, R = Republican, I = Independent",
     ylab="Second Dimension (Race / Civil Rights)",
     xlim=c(-1,1), ylim=c(-1,1), type="n")
points(wnom1[party=="D"], wnom2[party=="D"], pch="D", cex = 0.5, col="blue")
points(wnom1[party=="R"], wnom2[party=="R"], pch="R", cex = 0.5, col="red")
points(wnom1[party=="Indep"], wnom2[party=="Indep"], pch="I", cex = 0.5, col="black")
```



Compared to the map we have seen for the 108th House in class, the Republican block and the Democratic block are now further apart from one another. In particular, the Republican block has become more conservative while some Democrats still remain fairly close to the middle line 0.

The two blocks are separated primarily on the first dimension of ideology (progressive vs. conservative), and the separation is much less noticeable on the second dimension of various social issues/trends.

Dimensionality



We will primarily use visual diagnosis. The conclusion is that two dimensions is probably redundant to classify voting behavior in the 113th US House.

From the two plots on the right panel, we find that the cutting line angles are predominantly concentrated around 90, meaning that a great majority of voting for bills can be explained by just one dimension.

Looking at the Scree plot, the eigenvalues virtually vanish after dimension one, also indicating that dim=1 is probably a better option.

Discussion of Unfolding Methods

I have referenced to the following material:

Wiley Handbook of Psychometric Testing - Chapter 28 Psychometric Methods in Political Science.

The NOMINATE model assumes a Gaussian utility function, while the IRT model assumes a concave down quadratic utility function. In NOMINATE, if voting either Yea or Nay are far away from ideal point, the deterministic utility vanishes to 0 and the stochastic effect becomes significant. The IRT model assumes that the subject can always discern between the two alternatives - in fact, the further away the options are from their ideal point, the sharper their distinction, even with stochastic shock. One should think about the parametric utility function when choosing between NOMINATE and IRT.

The IRT model, when estimating multiple dimensions, becomes “compensatory”. In other words, it does not estimate the effects of two dimensions independently - it estiamtes the effect of multiple dimensions additively and interchangeably. In the case of political voting, if we encounter a legislator who is economically

conservative yet socially progressive, we should think carefully whether these two dimensions offset one another if we want to use the IRT method.

Optimal Classification is a non-parametric method and does not assume the form of utility function of error function. It does not seek to estimate parameters but instead tries to minimize classification error. Its results are probably less interpretable than the parametric unfolding methods, but it probably does better if we care more about prediction.