MRU: Analyses of Dutch Parliamentary Election data

Mark de Rooij

May 19, 2022

Abstract

Analysis of Dutch Parliamentary Election study data of 2002. There are five predictor variables: (E), Income Differences (ID), Asylum Seekers (AS), (C), and self left-right scaling (LR). The response variable is the vote in the 2002 election, it has originally 13 classes, but we removed classes with a frequency lower than 5, leaving 8 classes.

Data

```
load("~/surfdrive/LogitMDA/dpes02.Rdata")
mydat3= mydat2[, 14:19]
mydat3 = mydat3[complete.cases(mydat3), ]

X = mydat3[ , 1:5]; y = mydat3[, 6]
G = class.ind(y)
G = G[, c(1,2,3,4,5,7,9,10)]
y = y[which(rowSums(G) != 0)]
X = X[which(rowSums(G) != 0), ]
X = as.matrix(X)
X = X - outer(rep(1,nrow(X)), c(4,4,4,4,6))
G = G[which(rowSums(G) != 0), ]
colnames(G) = c("PvdA", "CDA", "VVD", "D66", "GL" , "CU", "LPF", "SP")

xnames = colnames(X)
ynames = colnames(G)
```

Investigating Local optima

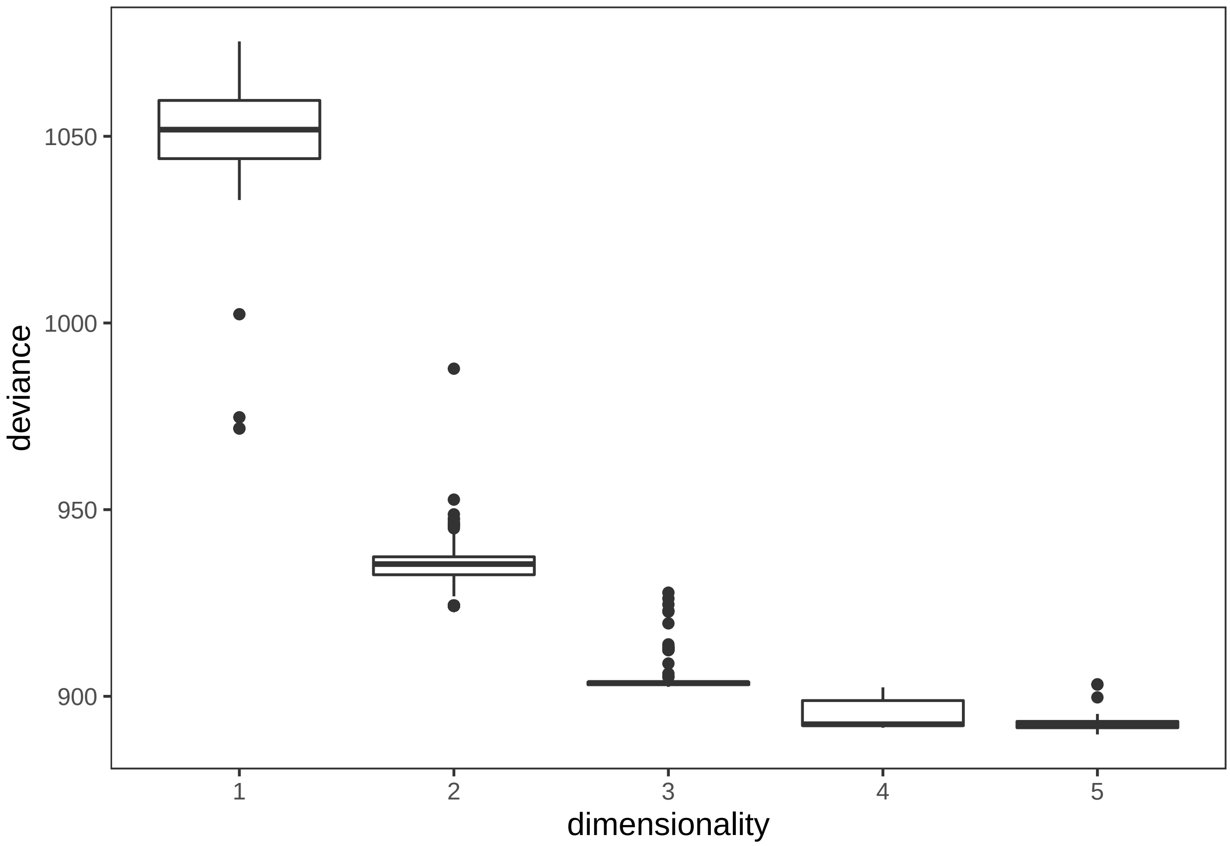
```
set.seed(1234)

M1results = vector(mode = "list", length = 100)
M2results = vector(mode = "list", length = 100)
M3results = vector(mode = "list", length = 100)
M4results = vector(mode = "list", length = 100)
M5results = vector(mode = "list", length = 100)
```

```r
M1results[[1]] = mru.da(X, G, m = 1)
M2results[[1]] = mru.da(X, G, m = 2)
M3results[[1]] = mru.da(X, G, m = 3)
M4results[[1]] = mru.da(X, G, m = 4)
M5results[[1]] = mru.da(X, G, m = 5)

for(r in 2:100){
  M1results[[r]] = mru.random( X, G, m = 1 )
  M2results[[r]] = mru.random( X, G, m = 2 )
  M3results[[r]] = mru.random( X, G, m = 3 )
  M4results[[r]] = mru.random( X, G, m = 4 )
  M5results[[r]] = mru.random( X, G, m = 5 )
}
save(M1results, M2results, M3results, M4results, M5results, file = "dpes02results.Rdata")
```

```r
library(ggplot2)
dev1 = rep(NA, 100)
dev2 = rep(NA, 100)
dev3 = rep(NA, 100)
dev4 = rep(NA, 100)
dev5 = rep(NA, 100)
for(r in 1:100){
  dev1[r] = M1results[[r]]$deviance
  dev2[r] = M2results[[r]]$deviance
  dev3[r] = M3results[[r]]$deviance
  dev4[r] = M4results[[r]]$deviance
  dev5[r] = M5results[[r]]$deviance
}
df = data.frame(repl = rep(1:100, 5), dimensionality = as.factor(rep(1:5, each = 100)), deviance

p = ggplot(df, aes(dimensionality, deviance)) + geom_boxplot() + theme_apa()
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/dpes02results.pdf", plot = p)
```

```
## Saving 6.5 x 4.5 in image
```

```r
p
```



Analysis with Distance Model

```
out.dpes1 = M1results[[which.min(dev1)]]
out.dpes2 = M2results[[which.min(dev2)]]
out.dpes3 = M3results[[which.min(dev3)]]
out.dpes4 = M4results[[which.min(dev4)]]
out.dpes5 = M5results[[which.min(dev5)]]

fit = matrix(NA, 5, 4)
fit[ , 1] = c(1,2,3, 4, 5)
fit[1, 2] = out.dpes1$deviance
fit[2, 2] = out.dpes2$deviance
fit[3, 2] = out.dpes3$deviance
fit[4, 2] = out.dpes4$deviance
fit[5, 2] = out.dpes5$deviance
fit[1, 3] = ncol(X) * 1 + ncol(G) * 1
fit[2, 3] = ncol(X) * 2 + ncol(G) * 2 - 2*1/2
fit[3, 3] = ncol(X) * 3 + ncol(G) * 3 - 3*2/2
fit[4, 3] = ncol(X) * 4 + ncol(G) * 4 - 4*3/2
fit[5, 3] = ncol(X) * 5 + ncol(G) * 5 - 5*4/2
fit[, 4] = fit[, 2] + 2* fit[, 3]
colnames(fit) = c("Dimensionality", "Deviance", "#params", "AIC")
```

```
fit
```

```
##      Dimensionality Deviance #params      AIC
## [1,]              1 971.7096      13 997.7096
## [2,]              2 924.2237      25 974.2237
## [3,]              3 902.5485      36 974.5485
## [4,]              4 891.5810      46 983.5810
## [5,]              5 889.7986      55 999.7986
```

Variable Selection

One by one leave out the predictor variables

```
BB = out.dpes2$B
VV = out.dpes2$V

out.dpes2.1 = mru.user(X[ , -1], G, m = 2, B.start = BB[-1, ], V.start = VV)
out.dpes2.2 = mru.user(X[ , -2], G, m = 2, B.start = BB[-2, ], V.start = VV)
out.dpes2.3 = mru.user(X[ , -3], G, m = 2, B.start = BB[-3, ], V.start = VV)
out.dpes2.4 = mru.da(X[ , -4], G, m = 2)
out.dpes2.5 = mru.da(X[ , -5], G, m = 2)


fit2 = matrix(NA, 6, 4)
fit2[ , 1] = c(0, 1, 2, 3, 4, 5)
fit2[1, 2] = out.dpes2$deviance
fit2[2, 2] = out.dpes2.1$deviance
fit2[3, 2] = out.dpes2.2$deviance
fit2[4, 2] = out.dpes2.3$deviance
fit2[5, 2] = out.dpes2.4$deviance
fit2[6, 2] = out.dpes2.5$deviance
fit2[1 , 3] = (ncol(X)) * 2 + ncol(G) * 2 - 2*1/2
fit2[2:6 , 3] = (ncol(X) -1) * 2 + ncol(G) * 2 - 2*1/2
fit2[ , 4] = fit2[, 2] + 2* fit2[, 3]
colnames(fit2) = c("Left out X", "Deviance", "#params", "AIC")
fit2
```

```
##      Left out X Deviance #params       AIC
## [1,]          0 924.2237      25  974.2237
## [2,]          1 940.2961      23  986.2961
## [3,]          2 929.1904      23  975.1904
## [4,]          3 934.0728      23  980.0728
## [5,]          4 930.2698      23  976.2698
## [6,]          5 968.0206      23 1014.0206
```

It seems all variables have a contribution.

Visualization

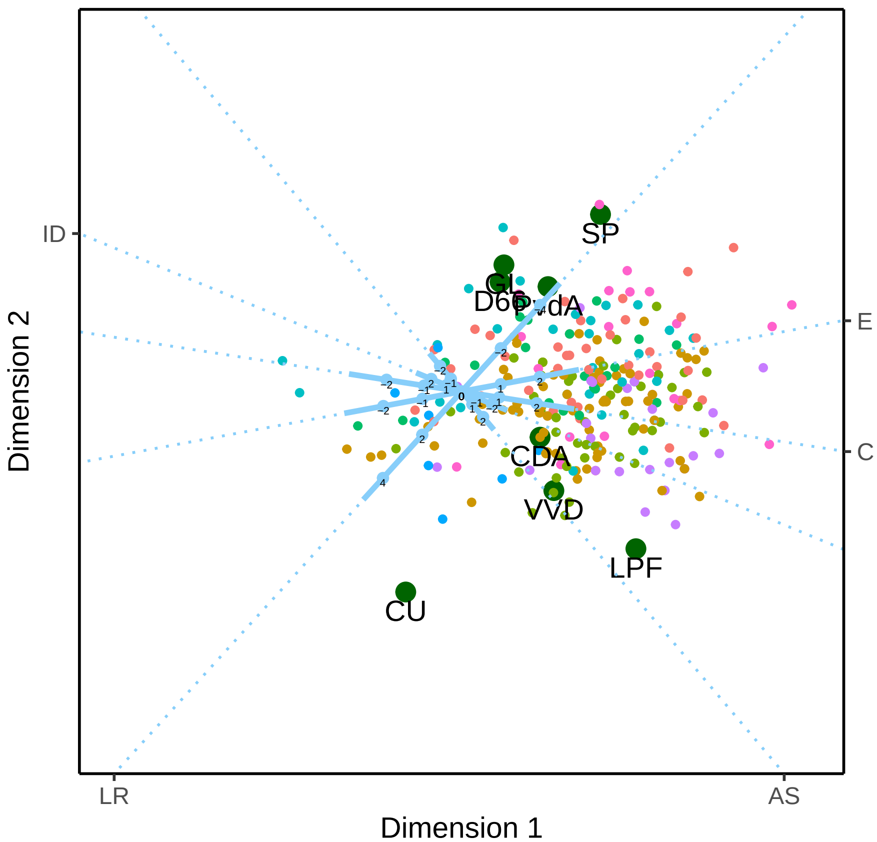
Let us have a closer look at the two dimensional solution.

```
##            [,1]       [,2]
## [1,]  0.6754028  0.1255911
## [2,] -0.2618483  0.1082304
## [3,]  0.1859372 -0.2202647
## [4,]  0.6483258 -0.1013019
## [5,] -0.3395111 -0.3734686
```

```
##            [,1]       [,2]
## [1,]  1.4928027  1.8118241
## [2,]  1.3550893 -0.7890429
## [3,]  1.5934964 -1.7109545
## [4,]  0.6651704  1.8908644
## [5,]  0.7311363  2.1871977
## [6,] -0.9646444 -3.4623975
## [7,]  3.0106207 -2.7149579
## [8,]  2.3991977  3.0579510
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in max(x): no non-missing arguments to max; returning -Inf
```




```
## Warning in min(x): no non-missing arguments to min; returning Inf
```

```
## Warning in min(x): no non-missing arguments to max; returning -Inf
```

Let us have a further look at the classification performance

```
D = outer(diag(U %*% t(U)), rep(1, 8)) + outer(rep(1,275), diag(V %*% t(V))) - 2* U %*% t(V)
D = sqrt(D)
PR = exp(-D)/rowSums(exp(-D))
yhat = apply(PR, 1, which.max)
yy = apply(G, 1, which.max)
tab1 = table(yy, yhat)
tab1
```

```
##     yhat
## yy    1  2  3  4  5  6  7  8
##   1 13 16  1  5  1  0  3  4
##   2  8 47  8  2  0  3  4  0
##   3  1 33 10  1  0  1  2  1
##   4  7 15  0  4  0  1  0  0
##   5  9 10  2  5  1  1  0  0
##   6  0  4  1  1  0  1  0  0
##   7  1 12  3  0  0  0 12  0
##   8  4  8  1  1  0  0  1  6
```

In total, 34.1818182 percent is correctly classified

Simulation

First develop a function that generates data based on population class points, regression weights and some covariance matrix:

```
gendata = function(N, covX, B, V){
  library(mvtnorm)
  library(poLCA)
  P = nrow(B)
  C = nrow(V)
  X = rmvnorm(N, mean = rep(0, P), sigma = covX)
  X = scale(X, center = TRUE, scale = FALSE)
  U = X %*% B

  D = sqrt(outer(diag(U %*% t(U)), rep(1, C)) + outer(rep(1, N), diag(V %*% t(V))) - 2 * U %*% t
  Pr = exp(-D)/rowSums(exp(-D))
  G = class.ind(rmulti(Pr))
  output = list(X = X, G = G)
}

mse = function(A, B){(A-B)^2}
```

And now we simulate data with these parameters and varying sample sizes (100, 200, 500, 1000). On the generated data sets we fit the multinomial restricted unfolding and we compare the obtained results with the population parameters.

```
V = out.dpes2$V
B = out.dpes2$B
covX = cov(X)

# small simulation
Ns = c(100, 200, 500, 1000)
plts = vector(mode = "list", length = length(Ns))
Bbias = vector(mode = "list", length = length(Ns))
Brmse = vector(mode = "list", length = length(Ns))
Vbias = vector(mode = "list", length = length(Ns))
Vrmse = vector(mode = "list", length = length(Ns))

source("ggbagplot.R")
set.seed(1234)
for(n in 1:length(Ns)){
  N = Ns[n]
  Bs = vector(mode = "list", length = 100)
  Vs = vector(mode = "list", length = 100)


  for(rep in 1:100){
    #cat("This is repetition:", rep, "for sample size", n ,"\n")
    mydat = gendata(N, covX, B, V)
    if(ncol(mydat$G) != 8){mydat = gendata(N, covX, B, V)}
    out <- mru.user(X = mydat$X, G = mydat$G, m = 2, B.start = B, V.start = V)
    # orthogonal procrustes analysis on U
    pq = svd(t(V) %*% out$V)
    TT = pq$v %*% t(pq$u)
    Bs[[rep]] = out$B %*% TT
    Vs[[rep]] = out$V %*% TT
  }

  # # bias
  Bbias[[n]] = Reduce("+", Bs) / length(Bs) - B
  Vbias[[n]] = Reduce("+", Vs) / length(Vs) - V
  #
  # rmse
  Brmse[[n]] = sqrt(Reduce("+" ,lapply(Bs, mse, B))/length(Bs))
  Vrmse[[n]] = sqrt(Reduce("+" ,lapply(Vs, mse, V))/length(Vs))

  #####################################################
  #####################################################
  # A visualization of the results
  #####################################################
  #####################################################
```

```r
Bdf = data.frame(B); colnames(Bdf) = c("x", "y")
Vdf = data.frame(V); colnames(Vdf) = c("x", "y")
Vdf$class = as.factor(c(1,2,3,4, 5, 6, 7, 8))

# class points
Vslong = matrix(NA, 800, 2); Bslong = matrix(NA, 500, 2)
for(r in 1:100){
  Vslong[((r-1)*8 + 1):(r*8), ] = Vs[[r]]
  Bslong[((r-1)*5 + 1):(r*5), ] = Bs[[r]]
}

Vslong = data.frame(cbind(rep(1:8, 100), Vslong))
colnames(Vslong) = c("class", "x", "y")
Vslong$class = as.factor(Vslong$class)

hull_data <-
  Vslong %>%
  group_by(class) %>%
  slice(chull(x, y))

# plt = ggplot(Vslong, aes(x = x, y = y, col = class)) +
#   # xlim(-15,15) +
#   # ylim(-15,15) +
#   geom_point(alpha = 0.5) +
#   geom_polygon(data = hull_data, aes(fill = class, colour = class), alpha = 0.3, show.legend
#   scale_color_manual(values = brewer.pal(9,"Greens")[6:9]) +
#   scale_fill_manual(values = brewer.pal(9,"Greens")[6:9])

plt = ggplot(Vslong, aes(x = x, y = y, col = class)) +
  geom_point(alpha = 0.5) +
  geom_bag(prop = 0.9, aes(fill = class, colour = class), alpha = 0.3, show.legend = FALSE) +
  scale_color_manual(values = brewer.pal(8,"Paired")[1:8]) +
  scale_fill_manual(values = brewer.pal(8,"Paired")[1:8]) +
  # scale_color_manual(values = brewer.pal(9,"Greens")[2:9]) +
  # scale_fill_manual(values = brewer.pal(9,"Greens")[2:9]) +
  theme(legend.position = "none")

# predictor variables
Bslong = data.frame(cbind(rep(1:5, 100), Bslong))
colnames(Bslong) = c("pred", "x", "y")
Bslong$pred = as.factor(Bslong$pred)

plt = plt +
  #geom_abline(intercept = 0, slope = Bslong$y/Bslong$x, col = "lightskyblue", alpha = 0.3) +
  geom_point(data = Bslong, aes(x = x, y = y), colour = "darkblue", alpha = 0.3)

plt = plt + geom_point(data = Vdf, aes(x = x, y = y), colour = brewer.pal(8,"Paired")[1:8],
```

```
                         shape = 18, size = 5) +
    geom_abline(intercept = 0, slope = Bdf[ ,2]/Bdf[ ,1], colour = "darkblue", size = 1) +
    geom_point(data = Bdf, aes(x =x, y = y), col = "darkblue", size = 5)

  plt = plt +
    labs(
      x = "Dimension 1",
      y = "Dimension 2"
      ) +
    xlim(-12,12) +
    ylim(-12,12) +
    coord_fixed() +
    theme_apa() +
    theme(legend.position = "none")

  # for(r in 1:100){
  #   Bdfr = data.frame(Bs[[r]]); colnames(Bdfr) = c("x", "y")
  #   Vdfr = data.frame(Vs[[r]]); colnames(Vdfr) = c("x", "y")
  #   plt = plt + geom_point(data = Bdfr, aes(x = x, y = y), col = "darkblue", size = 0.2, alpha
  #     geom_point(data = Vdfr, aes(x = x,y = y), col = "darkred", size = 0.2, alpha = 0.1)
  # }

  plts[[n]] = plt + labs(title = paste("Estimates for N = ", N))
}
```

```
## Loading required package: scatterplot3d

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
Bbias
```

```
## [[1]]
##             [,1]        [,2]
## [1,]  0.38903431  0.04276005
## [2,] -0.08482803  0.05831301
## [3,]  0.09963935 -0.13185406
## [4,]  0.33929648 -0.13517173
## [5,] -0.13071999 -0.13458402
##
## [[2]]
##             [,1]          [,2]
## [1,]  0.10423514  0.0479936641
## [2,] -0.07564222  0.0002939442
```

```
## [3,]  0.06724799 -0.0521841627
## [4,]  0.18807728 -0.0050447557
## [5,] -0.08099885 -0.0713661095
##
## [[3]]
##             [,1]         [,2]
## [1,]  0.05208296  0.017930480
## [2,] -0.04217254  0.009849422
## [3,]  0.04479090 -0.006855323
## [4,]  0.06884586 -0.006724797
## [5,] -0.02603851 -0.026569416
##
## [[4]]
##              [,1]         [,2]
## [1,] -0.014229641  0.005599010
## [2,]  0.008318828  0.005206908
## [3,]  0.002300260 -0.009150634
## [4,] -0.004745253 -0.002195368
## [5,]  0.001950599 -0.010585330
```

```
Brmse
```

```
## [[1]]
##           [,1]      [,2]
## [1,] 0.9016249 0.2863115
## [2,] 0.5566005 0.2352794
## [3,] 1.0541109 0.3029138
## [4,] 1.1313677 0.3899638
## [5,] 0.6156543 0.2538901
##
## [[2]]
##           [,1]      [,2]
## [1,] 0.4019670 0.1336394
## [2,] 0.3192927 0.1208003
## [3,] 0.3728930 0.1476837
## [4,] 0.5954733 0.1752737
## [5,] 0.3396462 0.1336104
##
## [[3]]
##           [,1]       [,2]
## [1,] 0.1864330 0.06467349
## [2,] 0.1544163 0.05818670
## [3,] 0.1732663 0.06632782
## [4,] 0.2505395 0.07648043
## [5,] 0.1301725 0.06810153
##
## [[4]]
##            [,1]       [,2]
## [1,] 0.09004273 0.03752315
```

```
## [2,] 0.08273703 0.03631230
## [3,] 0.09730373 0.04124534
## [4,] 0.11618547 0.05520182
## [5,] 0.07308860 0.03823882
```

```
Vbias
```

```
## [[1]]
##              [,1]       [,2]
## [1,]  0.682661633  0.2673623
## [2,]  0.615193605 -0.2479751
## [3,]  0.527629500 -0.5055585
## [4,]  0.544070967  0.2190757
## [5,]  0.535437201  0.3012005
## [6,] -0.005630273 -1.0795659
## [7,]  1.255473712 -0.4577367
## [8,]  0.852648502  0.8253223
##
## [[2]]
##           [,1]       [,2]
## [1,] 0.3105929  0.2333157
## [2,] 0.3585781 -0.1721241
## [3,] 0.3930953 -0.2152490
## [4,] 0.2073508  0.2797443
## [5,] 0.3011406  0.2443164
## [6,] 0.1674000 -0.5549939
## [7,] 0.3931247 -0.3241263
## [8,] 0.4635397  0.3051915
##
## [[3]]
##            [,1]        [,2]
## [1,] 0.02271760  0.11989120
## [2,] 0.13539981 -0.01980091
## [3,] 0.12996460 -0.03892398
## [4,] 0.04781049  0.09627779
## [5,] 0.08987820  0.09234049
## [6,] 0.04331010 -0.15993811
## [7,] 0.10655697 -0.20030966
## [8,] 0.15910734  0.11392757
##
## [[4]]
##              [,1]         [,2]
## [1,]  0.006210809  0.021144013
## [2,]  0.010249641  0.006366099
## [3,]  0.026928826 -0.032823611
## [4,] -0.013713372  0.023771791
## [5,] -0.010356077  0.033385827
## [6,] -0.110670944 -0.022099488
## [7,]  0.137913068  0.052011447
```

```
## [8,]  0.060921943 -0.042700695
```

```
Vrmse
```

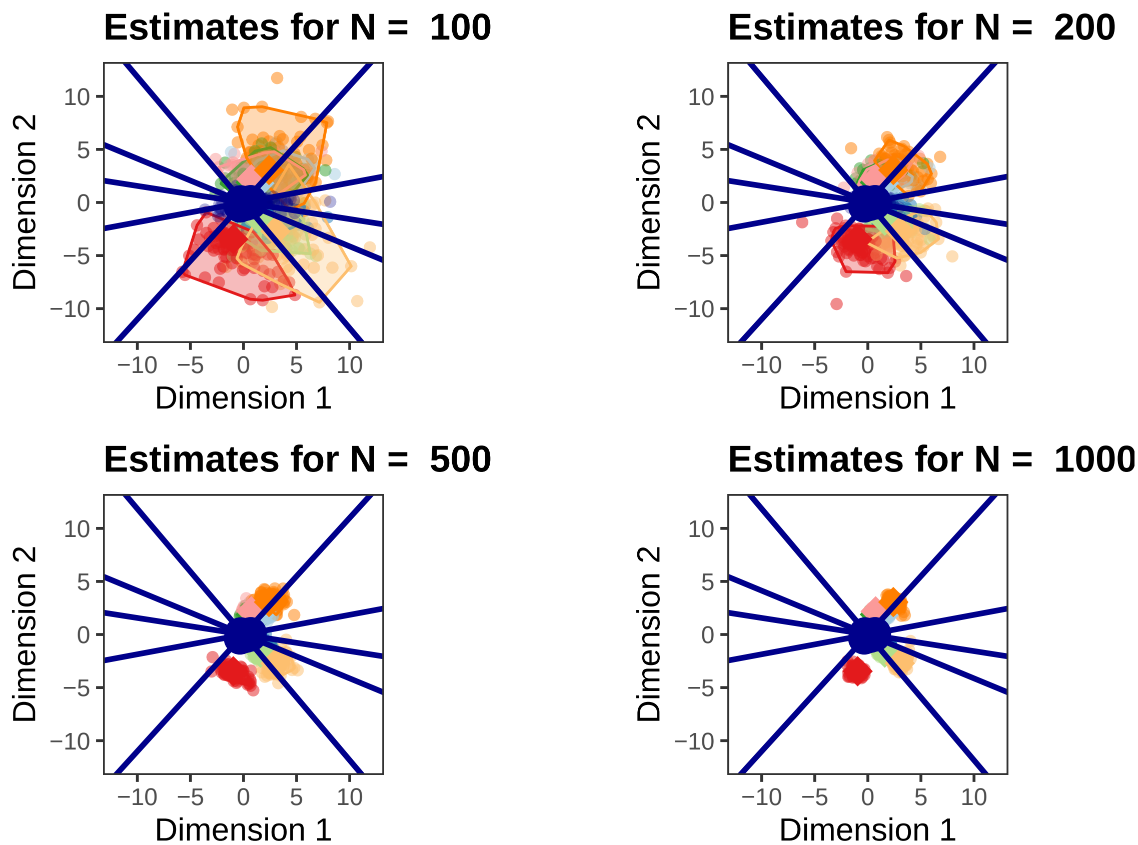
```
## [[1]]
##          [,1]      [,2]
## [1,] 1.738198 1.2624988
## [2,] 1.520872 0.8723253
## [3,] 1.612636 1.2828335
## [4,] 1.780707 1.2185258
## [5,] 1.870530 1.2006336
## [6,] 2.185557 2.1967881
## [7,] 2.733648 2.3797338
## [8,] 2.102379 2.2699087
##
## [[2]]
##          [,1]      [,2]
## [1,] 1.076238 0.6233200
## [2,] 1.045410 0.5898637
## [3,] 1.107696 0.6378368
## [4,] 1.087498 0.6792636
## [5,] 1.117317 0.6722571
## [6,] 1.459634 1.3058306
## [7,] 1.467694 1.1231800
## [8,] 1.390710 1.1821296
##
## [[3]]
##           [,1]      [,2]
## [1,] 0.5513351 0.3564071
## [2,] 0.4240130 0.3141090
## [3,] 0.4636461 0.3735574
## [4,] 0.4937967 0.3072559
## [5,] 0.5197153 0.3368310
## [6,] 0.7410864 0.5163434
## [7,] 0.6219503 0.6546733
## [8,] 0.6727319 0.5138417
##
## [[4]]
##           [,1]      [,2]
## [1,] 0.2416031 0.1765487
## [2,] 0.2091087 0.1568418
## [3,] 0.2401074 0.1897934
## [4,] 0.2264376 0.1895712
## [5,] 0.2564851 0.1780745
## [6,] 0.3676040 0.3037857
## [7,] 0.3850063 0.4197027
## [8,] 0.3993257 0.3566329
```

```
save(Bbias, Brmse, Vbias, Vrmse, file = "simdpesresults.Rdata")
plt = ggarrange(plts[[1]], plts[[2]], plts[[3]], plts[[4]], nrow = 2, ncol = 2)
```

```
## Warning: Removed 3 rows containing non-finite values (statbag).
```

```
## Warning: Removed 3 rows containing missing values (geom_point).
```

```
plt
```




```
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/simdpes.pdf", plot = plt, width = 11.7, height =
```

Analysis with Squared Distance Model

Identification restrictions

- Translation: set the coordinates for first class to zero
- Rotation: set the upper triangle of **V to zero**
- **Scaling: set some elements in V to one**

The following function minimizes the deviance for the IPC model.

With the following code we first create our design matrix and initiate parameters for minimization of the deviance function. Then we call the `optim`-function to find a set of parameters that maximizes the likelihood or minimizes the deviance.

```
da.out = mru.start(X, G, m = 2, start = "da")
X = cbind(1,X)
#pars0 = c(rep(0,12), rnorm(13))
pars0 = c(0, da.out$B[,1], 0, da.out$B[,2], da.out$V[2:8, 1], da.out$V[3:8, 2])
out = optim(pars0,ipc.deviance, G = G, X = X, method = "BFGS", control = list(trace = 2, maxit =
```

```
## initial  value 1205.424053
## iter  10 value 985.445622
## iter  20 value 967.327201
## iter  30 value 945.492090
## iter  40 value 940.696134
## iter  50 value 934.737638
## iter  60 value 934.645110
## iter  70 value 934.490096
## iter  80 value 934.326628
## iter  90 value 934.211804
## iter 100 value 934.165971
## iter 110 value 934.147458
## iter 120 value 934.137176
## iter 130 value 934.113742
## iter 140 value 934.079854
## iter 150 value 934.065999
## iter 160 value 934.061531
## iter 160 value 934.061528
## final  value 934.061251
## converged
```

The value of the deviance is 934.0612509 for which the estimated parameters are

```
P = ncol(X)
C = ncol(G)

B = matrix(out$par[1:(2*P)], P, 2)
B
```

```
##             [,1]         [,2]
## [1,] -2.2775977  0.655256941
## [2,]  0.5156497 -0.182406729
## [3,]  0.3727131  0.019235423
## [4,] -0.8209635  0.002658173
## [5,] -0.2081842 -0.102692906
## [6,] -1.6292980  0.153803216
```

```
U = X %*% B

V = matrix(0, C, 2)
V[2:C,1] = out$par[(2*P+1):(2*P+7)]
V[3:C,2] = out$par[(2*P+8):(2*P+13)]
```

```
V
```

```
##              [,1]       [,2]
## [1,]  0.00000000  0.0000000
## [2,] -0.15594058  0.0000000
## [3,] -0.20813499 -0.5548090
## [4,]  0.03772892  0.8089669
## [5,]  0.06583659  0.8128167
## [6,] -0.10895865  2.1168628
## [7,] -0.31141406 -1.2135520
## [8,] -0.02775300 -0.8416976
```

Let us have a further look at the classification performance

```
D2 = outer(diag(U %*% t(U)), rep(1, 8)) + outer(rep(1,275), diag(V %*% t(V))) - 2* U %*% t(V)
PR2 = exp(-D2)/rowSums(exp(-D2))
yhat2 = apply(PR2, 1, which.max)
tab2 = table(yy, yhat2)
tab2
```

```
##    yhat2
## yy   1  2  3  4  5  6  7  8
##   1 17 15  5  1  4  0  0  1
##   2 11 51  4  0  0  3  3  0
##   3  6 35  6  0  0  1  1  0
##   4  8 16  0  0  2  1  0  0
##   5 10 10  2  1  4  1  0  0
##   6  0  6  0  0  1  0  0  0
##   7  1 14  5  0  0  0  7  1
##   8  8  7  2  0  1  0  1  2
```

In total, 31.6363636 percent is correctly classified

Graphical representation

Let us have a further look at the classification performance

```
D2 = outer(diag(UU %*% t(UU)), rep(1, 8)) + outer(rep(1,275), diag(VV %*% t(VV))) - 2* UU %*% t(
PR2 = exp(-D2)/rowSums(exp(-D2))
yhat2 = apply(PR2, 1, which.max)
tab2 = table(yy, yhat2)
tab2
```

```
##    yhat2
## yy   1  2  3  4  5  6  7  8
##   1 17 15  5  1  4  0  0  1
##   2 11 51  4  0  0  3  3  0
##   3  6 35  6  0  0  1  1  0
##   4  8 16  0  0  2  1  0  0
##   5 10 10  2  1  4  1  0  0
##   6  0  6  0  0  1  0  0  0
```

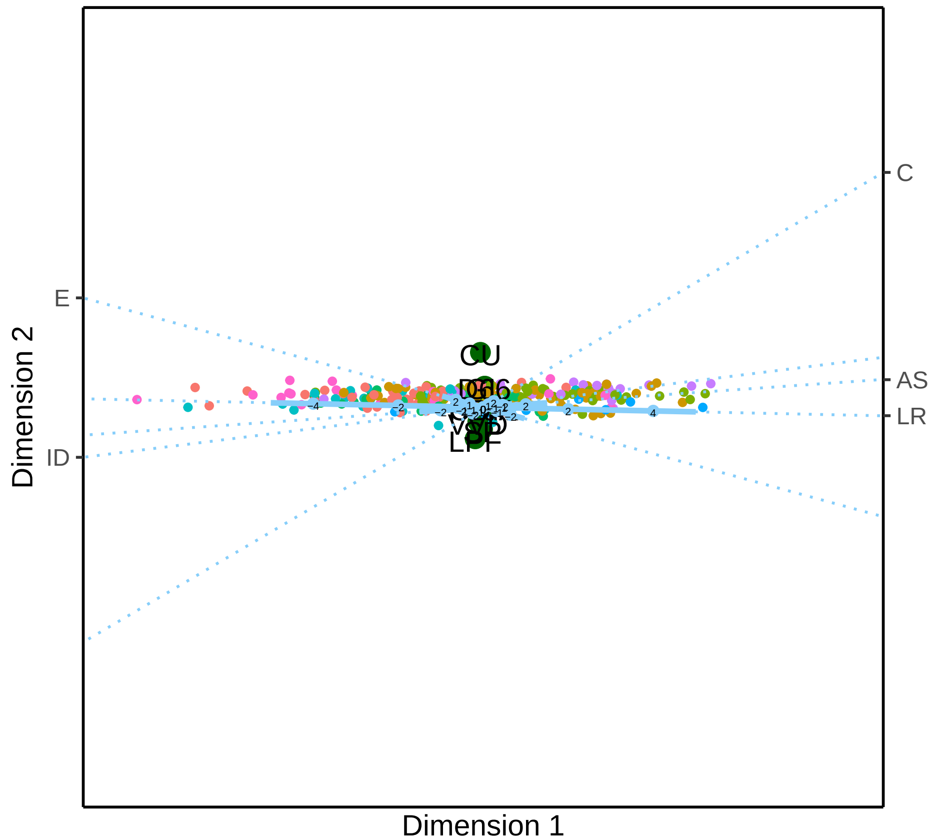
```
##    7   1 14   5   0   0   0   7   1
##    8   8   7   2   0   1   0   1   2
```

In total, 31.6363636 percent is correctly classified

```
NN = as.data.frame(UU)
colnames(NN) = c("Dim1", "Dim2")

VV = as.data.frame(V)
colnames(VV) = c("Dim1", "Dim2")
P = nrow(B)
Xo = X[, -1]

# for solid line
MCx1 <- data.frame(labs=character(),
                   varx = integer(),
                   Dim1 = double(),
                   Dim2 = double(), stringsAsFactors=FALSE)
# for markers
MCx2 <- data.frame(labs=character(),
                   varx = integer(),
                   Dim1 = double(),
                   Dim2 = double(), stringsAsFactors=FALSE)

ll = 0
lll = 0
for(pp in 1:P){
  b = matrix(B[pp , ], 2, 1)
  # solid line
  minx = min(Xo[, pp])
  maxx = max(Xo[, pp])
  m.x1 = c(minx,maxx)
  markers1 = matrix(m.x1, 2, 1)
  markerscoord1 = outer(markers1, b) # markers1 %*% t(b %*% solve(t(b) %*% b))
  MCx1[(ll + 1): (ll + 2), 1] = paste0(c("min", "max"), pp)
  MCx1[(ll + 1): (ll + 2), 2] = pp
  MCx1[(ll + 1): (ll + 2), 3:4] = markerscoord1
  ll = ll + 2
  # markers
  m.x2 = pretty(Xo[, pp])
  m.x2 = m.x2[which(m.x2 > minx & m.x2 < maxx)]
  l.m = length(m.x2)
  markers2 = matrix(m.x2, l.m, 1)
  markerscoord2 = outer(markers2, b) # markers2 %*% t(b %*% solve(t(b) %*% b))
  MCx2[(lll + 1): (lll + l.m), 1] = paste(m.x2)
  MCx2[(lll + 1): (lll + l.m), 2] = pp
  MCx2[(lll + 1): (lll + l.m), 3:4] = markerscoord2
  lll = lll + l.m
```

```
} # loop p

p2 = ggplot() +
    geom_point(data = VV, aes(x = Dim1, y = Dim2), colour = "darkgreen", size = 3) +
    geom_point(data = NN, aes(x = Dim1, y = Dim2, color = y), size = 1, show.legend = FALSE) +
    xlab("Dimension 1") +
    ylab("Dimension 2")

p2 = p2 + geom_text(data = VV, aes(x = Dim1, y = Dim2),
                  label = colnames(G),
                  vjust = 0, nudge_y = -0.5)

xcol = "lightskyblue"
p2 = p2 + geom_abline(intercept = 0, slope = B[,2]/B[,1], colour = xcol, linetype = 3) +
    geom_line(data = MCx1, aes(x = Dim1, y = Dim2, group = varx), col = xcol, size = 1) +
    geom_point(data = MCx2, aes(x = Dim1, y = Dim2), col = xcol) +
    geom_text(data = MCx2, aes(x = Dim1, y = Dim2, label = labs), nudge_y = -0.08, size = 1.5)

a = ceiling(max(abs(c(ggplot_build(p2)$layout$panel_scales_x[[1]]$range$range, ggplot_build(p2)$

idx1 = apply(abs(B), 1, which.max)
t = s = rep(NA,(P))
for(pp in 1:(P)){
    t[(pp)] = (a *1.1)/(abs(B[pp,idx1[(pp)]])) * B[pp,-idx1[(pp)]]
    s[(pp)] = sign(B[pp,idx1[(pp)]])
}
CC = cbind(idx1, t, s)
bottom = which(CC[, "idx1"] == 2 & CC[, "s"] == -1)
top =  which(CC[, "idx1"] == 2 & CC[, "s"] == 1)
right = which(CC[, "idx1"] == 1 & CC[, "s"] == 1)
left = which(CC[, "idx1"] == 1 & CC[, "s"] == -1)

p2 = p2 + scale_x_continuous(limits = c(-a,a), breaks = CC[bottom, "t"], labels = xnames[bottom]
                             sec.axis = sec_axis(trans ~ ., breaks = CC[top, "t"], labels = xnam
p2 = p2 + scale_y_continuous(limits = c(-a,a), breaks = CC[left, "t"], labels = xnames[left],
                             sec.axis = sec_axis(trans ~ ., breaks = CC[right, "t"], labels = xn
p2 = p2 + coord_fixed()

p2 = p2 + theme_bw()
p2 = p2 + theme(axis.line = element_line(colour = "black"),
                panel.grid.major = element_blank(),
                panel.grid.minor = element_blank(),
                panel.border = element_blank(),
                panel.background = element_blank())



p2
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf
## Warning in max(x): no non-missing arguments to max; returning -Inf
```




```
ggsave("~/surfdrive/multldm/mrmdu/paper/figures/dpesplot2.pdf", plot = p2, width = 11.7, height
```

```
## Warning in min(x): no non-missing arguments to min; returning Inf

## Warning in min(x): no non-missing arguments to max; returning -Inf
```

Multinomial Logistic Regression

As a comparison we also fit a standard multinomial logistic regression on this data set. Here is the R-function for minimizing the multinomial deviance again.

We use the first catgeory as baseline and fit the model with the following code:

```
pars0 = matrix(0,42,1)
out.mlr <- optim(pars0, multinomial.deviance, NULL, Y = G, X = X, method="BFGS",control = list(t
```

```
## initial  value 1143.692848
## iter  10 value 943.753513
## iter  20 value 926.164197
## iter  30 value 916.123032
## iter  40 value 912.983640
## iter  50 value 912.565456
```

```
## iter  50 value 912.565450
## iter  50 value 912.565450
## final  value 912.565450
## converged
```

The deviance is 912.56545 which is lower, but not substantially than the deviance of the ideal point model 934.0612509. In this case the number of parameters of the multinomial logistic regression model is 12, while for the ideal point model it is 11.

```
B = matrix(out.mlr$par,ncol(X),ncol(G)-1)
colnames(B) = c('CDA/PvdA', 'VVD/PvdA', 'D66/PvdA', "GL/PvdA", "CU/PvdA", "LPF/PvdA", "SP/PvdA")
rownames(B) = c('Intercept', colnames(X)[-1])
knitr::kable(B, digits = 2, caption = 'Estimated parameter values')
```

Table 1: Estimated parameter values

	CDA/PvdA	VVD/PvdA	D66/PvdA	GL/PvdA	CU/PvdA	LPF/PvdA	SP/PvdA
Intercept	0.94	0.14	0.51	0.54	-1.86	-1.56	-1.65
E	-0.07	-0.03	-0.05	-0.12	-1.04	0.45	0.20
ID	-0.09	-0.28	0.04	-0.01	-0.09	-0.28	-0.04
AS	0.35	0.14	-0.09	-0.02	0.39	0.74	0.25
C	-0.15	0.14	-0.45	-0.48	-0.14	-0.08	0.10
LR	0.54	0.64	0.30	0.11	0.80	0.64	-0.26

Using the same procedure as before we can obtain standard errors, z-statistics and p-values for each of the parameter estimates.

```
SEs = sqrt(diag(solve(out.mlr$hessian)))
zstats = out.mlr$par/SEs
pvals = 1 - pnorm(abs(zstats))
TAB = cbind(out.mlr$par, SEs, zstats, pvals)
colnames(TAB) = c("estimates", "stand.error", "z", "p")
knitr::kable(TAB, digits = 2, caption = 'Estimates and Standard Errors')
```

Table 2: Estimates and Standard Errors

estimates	stand.error	z	p
0.94	0.34	2.79	0.00
-0.07	0.09	-0.78	0.22
-0.09	0.11	-0.79	0.21
0.35	0.13	2.74	0.00
-0.15	0.15	-1.02	0.15
0.54	0.10	5.38	0.00
0.14	0.40	0.35	0.36
-0.03	0.10	-0.24	0.40
-0.28	0.12	-2.37	0.01
0.14	0.14	0.97	0.17
0.14	0.17	0.81	0.21

estimates	stand.error	z	p
0.64	0.11	5.83	0.00
0.51	0.36	1.41	0.08
-0.05	0.12	-0.40	0.34
0.04	0.14	0.27	0.39
-0.09	0.15	-0.57	0.28
-0.45	0.16	-2.78	0.00
0.30	0.12	2.45	0.01
0.54	0.36	1.49	0.07
-0.12	0.11	-1.04	0.15
-0.01	0.14	-0.04	0.48
-0.02	0.15	-0.15	0.44
-0.48	0.16	-3.02	0.00
0.11	0.12	0.87	0.19
-1.86	0.75	-2.49	0.01
-1.04	0.21	-4.95	0.00
-0.09	0.20	-0.43	0.33
0.39	0.24	1.63	0.05
-0.14	0.28	-0.51	0.30
0.80	0.21	3.80	0.00
-1.56	0.54	-2.89	0.00
0.45	0.16	2.75	0.00
-0.28	0.14	-2.02	0.02
0.74	0.18	4.07	0.00
-0.08	0.21	-0.37	0.36
0.64	0.13	5.00	0.00
-1.65	0.50	-3.29	0.00
0.20	0.14	1.45	0.07
-0.04	0.15	-0.28	0.39
0.25	0.16	1.57	0.06
0.10	0.19	0.54	0.29
-0.26	0.13	-2.07	0.02