Package `mactivate`
Examples I

Dave Zes

March 2, 2021

1 Example: Small Data, Categorical Inputs, Numerical Response

```
library(mactivate)
set.seed(777)
f_dummy <- function(x) {
      u.x <- sort(unique(x))
      d <- length(u.x)
      mx.out <- matrix(0, length(x), d)
      for (i in 1:d) {
          mx.out[, i] <- as.integer(u.x[i] == x)
      }
      colnames(mx.out) <- u.x
      return(mx.out)
  }
## small

N <- 80000
x1_dom <- toupper(letters[1:5])
x1 <- sample( x1_dom, size=N, replace=TRUE )
x2_dom <- paste0("x", 1:10)
x2 <- sample( x2_dom, size=N, replace=TRUE )
x3_dom <- c("Yes", "No", "NR")
x3 <- sample( x3_dom, size=N, replace=TRUE )
aX1 <- f_dummy(x1)
aX2 <- f_dummy(x2)
```

```
aX3 <- f_dummy(x3)
X <- cbind(aX1, aX2, aX3)
d <- ncol(X)
dim(X)
b <- rep_len( c(-2, 2), d )
ystar <-
  X %*% b +
  10 * X[ , "A"] * X[ , "Yes"] -
  10 * X[ , "B"] * X[ , "No"]
xtrue_formula <- eval(parse(text="y ~ . + x1:x3"))
m_tot <- 5
Xall <- t( ( t(X) - apply(X, 2, mean) ) / apply(X, 2, sd) )
#Xall <- 2 * X - 1

#Xall <- X + rnorm(prod(dim(X)), 0, 1/10)




xnoint_formula <- eval(parse(text="y ~ ."))
errs <- rnorm(N, 0, 5)
y <- ystar + errs
Nall <- N
cov(X)
yall <- y
sd(y)
dfx <- data.frame("y"=yall, x1, x2, x3)
#################### incorrectly fit LM: no interactions

xlm <- lm(xnoint_formula , data=dfx)
summary(xlm)
yhat <- predict(xlm, newdata=dfx)
sqrt( mean( (yall - yhat)^2 ) )
#################### incorrectly fit LM: no lower-order interactions
xlm <- lm(xtrue_formula , data=dfx)
summary(xlm)
yhat <- predict(xlm, newdata=dfx)
sqrt( mean( (yall - yhat)^2 ) )
xxfoldNumber <- rep_len( 1:4, Nall )
```

```
ufolds <- sort(unique(xxfoldNumber))
######################


xndx_test <- xxfoldNumber %in% 1
xndx_train <- setdiff( 1:Nall, xndx_test )
###################

X_train <- Xall[ xndx_train, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_train <- yall[ xndx_train ]
y_test <- yall[ xndx_test ]
###################


############################################################ hybrid
############################################################ hybrid

##### takes about 10 minutes


xcmact_hybrid <-
  f_control_mactivate(
  param_sensitivity = 10^11,
  bool_free_w = TRUE,
  w0_seed = 0.05,
  w_col_search = "alternate",
  max_internal_iter = 500,
  ss_stop = 10^(-12),
  escape_rate = 1.005,
  Wadj = 1/1,
  force_tries = 0,
  lambda = 1/1000,
  tol = 10^(-12)
  )
Uall <- Xall
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
```

```
X_train <- Xall[ xndx_train, , drop=FALSE ]
y_train <- yall[ xndx_train ]
xxnow <- Sys.time()
xxls_out <-
  f_fit_hybrid_01(
  X = X_train,
  y = y_train,
  m_tot = m_tot,
  U = X_train,
  m_start = 1,
  mact_control = xcmact_hybrid,
  verbosity = 1
  )
cat( difftime(Sys.time(), xxnow, units="mins"), "\n" )
######### check test error

U_test <- Xall[ xndx_test, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_test <- yall[ xndx_test ]
yhatTT <- matrix(NA, length(xndx_test), m_tot+1)
for(iimm in 0:m_tot) {
      yhat_fold <- predict(object=xxls_out, X0=X_test, U0=U_test, mcols=iimm )
      yhatTT[ , iimm + 1 ] <- yhat_fold
  }
errs_by_m <- NULL
for(iimm in 1:ncol(yhatTT)) {
      yhatX <- yhatTT[ , iimm]
      errs_by_m[ iimm ] <- sqrt(mean( (y_test - yhatX)^2 ))
      cat(iimm, "::", errs_by_m[ iimm ])
  }
##### plot test RMSE vs m

plot(0:(length(errs_by_m)-1), errs_by_m, type="l", xlab="m", ylab="RMSE Cost")
################## MLR test using 'correct' model
xtrue_formula_use <- xtrue_formula
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
xlm <- lm(xtrue_formula_use , data=dfx[ xndx_train, ])
```

```
yhat <- predict(xlm, newdata=dfx[ xndx_test, ])
sqrt( mean( (y_test - yhat)^2 ) )
###################################  gradient
###################################  gradient

##### takes about 15-30 minutes


#Xall <- t( ( t(X) - apply(X, 2, mean) ) / apply(X, 2, sd) )

#Xall <- 2 * X - 1

set.seed(777)
Xall <- t( ( t(X) - apply(X, 2, mean) ) / apply(X, 2, sd) ) + rnorm(prod(dim(X)), 0, 1/50)
xcmact_gradient <-
  f_control_mactivate(
  param_sensitivity = 10^14,
  bool_free_w = TRUE,
  w0_seed = 0.1,
  w_col_search = "alternate",
  bool_headStart = TRUE,
  ss_stop = 10^(-18), ### very small
  escape_rate = 1.005,
  step_size = 1,
  Wadj = 1/1,
  force_tries = 100,
  lambda = 0
  )
#### Fit


Uall <- Xall
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
X_train <- Xall[ xndx_train, , drop=FALSE ]
y_train <- yall[ xndx_train ]
xxnow <- Sys.time()
xxls_out <-
```

```
  f_fit_gradient_01(
  X = X_train,
  y = y_train,
  m_tot = m_tot,
  U = X_train,
  m_start = 1,
  mact_control = xcmact_gradient,
  verbosity = 1
  )
cat( difftime(Sys.time(), xxnow, units="mins"), "\n" )
######### check test error

U_test <- Xall[ xndx_test, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_test <- yall[ xndx_test ]
yhatTT <- matrix(NA, length(xndx_test), m_tot+1)
for(iimm in 0:m_tot) {
      yhat_fold <- predict(object=xxls_out, X0=X_test, U0=U_test, mcols=iimm )
      yhatTT[ , iimm + 1 ] <- yhat_fold
  }
errs_by_m <- NULL
for(iimm in 1:ncol(yhatTT)) {
      yhatX <- yhatTT[ , iimm]
      errs_by_m[ iimm ] <- sqrt(mean( (y_test - yhatX)^2 ))
      cat(iimm, "::", errs_by_m[ iimm ])
  }
##### plot test RMSE vs m

plot(0:(length(errs_by_m)-1), errs_by_m, type="l", xlab="m", ylab="RMSE Cost")
```

2 Example: Small Data, Categorical Inputs, Dichotomous Response

```
library(mactivate)
set.seed(777)
f_dummy <- function(x) {
      u.x <- sort(unique(x))
```

```
      d <- length(u.x)
      mx.out <- matrix(0, length(x), d)
      for (i in 1:d) {
          mx.out[, i] <- as.integer(u.x[i] == x)
      }
      colnames(mx.out) <- u.x
      return(mx.out)
  }
## small

N <- 80000
x1_dom <- toupper(letters[1:5])
x1 <- sample( x1_dom, size=N, replace=TRUE )
x2_dom <- paste0("x", 1:10)
x2 <- sample( x2_dom, size=N, replace=TRUE )
x3_dom <- c("Yes", "No", "NR")
x3 <- sample( x3_dom, size=N, replace=TRUE )
aX1 <- f_dummy(x1)
aX2 <- f_dummy(x2)
aX3 <- f_dummy(x3)
X <- cbind(aX1, aX2, aX3)
d <- ncol(X)
dim(X)
b <- rep_len( c(-2/3, 2/3), d )
ystar <-
  X %*% b +
  7 * X[ , "A"] * X[ , "Yes"] -
  7 * X[ , "B"] * X[ , "No"]
xtrue_formula <- eval(parse(text="y ~ . + x1:x3"))
ysigmoid <- 1 / (1 + exp(-ystar))
range(ysigmoid)
y <- rbinom(size=1, n=N ,prob=ysigmoid)
m_tot <- 5
xnoint_formula <- eval(parse(text="y ~ ."))
Nall <- N
cov(X)
yall <- y
######## standardize and add a smidge of noise to inputs
```

```
set.seed(777)
Xall <- t( ( t(X) - apply(X, 2, mean) ) / apply(X, 2, sd) ) + rnorm(prod(dim(X)), 0, 1/40)
dfx <- data.frame("y"=yall, x1, x2, x3)
################### incorrectly fit LM: no interactions
xglm <- glm(xnoint_formula , data=dfx, family=binomial(link="logit"))
summary(xglm)
yhat <- predict(xglm, newdata=dfx, type="response")
mean( f_logit_cost(y=yall, yhat=yhat) )
####### known true
xglm <- glm(xtrue_formula , data=dfx, family=binomial(link="logit"))
summary(xglm)
yhat <- predict(xglm, newdata=dfx, type="response")
mean( f_logit_cost(y=yall, yhat=yhat) )
xxfoldNumber <- rep_len( 1:4, Nall )
ufolds <- sort(unique(xxfoldNumber))
######################


xndx_test <- xxfoldNumber %in% 1
xndx_train <- setdiff( 1:Nall, xndx_test )
###################

X_train <- Xall[ xndx_train, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_train <- yall[ xndx_train ]
y_test <- yall[ xndx_test ]
###################

########## descent is very slow at first
##### takes about 30 minutes

xcmact_gradient <-
  f_control_mactivate(
  param_sensitivity = 10^14,
  bool_free_w = TRUE,
  w0_seed = 0.05,
  w_col_search = "alternate",
  bool_headStart = TRUE,
  ss_stop = 10^(-18), ### very small
```

```
  escape_rate = 1.005,
  step_size = 1,
  Wadj = 1/1,
  force_tries = 0,
  lambda = 1/1 #### does nothing here
  )
#### Fit


Uall <- Xall
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
X_train <- Xall[ xndx_train, , drop=FALSE ]
y_train <- yall[ xndx_train ]
xxnow <- Sys.time()
xxls_out <-
  f_fit_gradient_logistic_01(
  X = X_train,
  y = y_train,
  m_tot = m_tot,
  U = X_train,
  m_start = 1,
  mact_control = xcmact_gradient,
  verbosity = 1
  )
cat( difftime(Sys.time(), xxnow, units="mins"), "\n" )
######### check test error

U_test <- Xall[ xndx_test, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_test <- yall[ xndx_test ]
yhatTT <- matrix(NA, length(xndx_test), m_tot+1)
for(iimm in 0:m_tot) {
      yhat_fold <- predict(object=xxls_out, X0=X_test, U0=U_test, mcols=iimm )
      yhatTT[ , iimm + 1 ] <- yhat_fold[[ "p0hat" ]]
  }
errs_by_m <- NULL
for(iimm in 1:ncol(yhatTT)) {
```

```
      yhatX <- yhatTT[ , iimm]
      errs_by_m[ iimm ] <- mean( f_logit_cost(y=y_test, yhat=yhatX) )
      cat(iimm, "::", errs_by_m[ iimm ])
  }
##### plot test Logit Cost vs m

plot(0:(length(errs_by_m)-1), errs_by_m, type="l", xlab="m", ylab="Logit Cost")
################## test off 'correct' model
xtrue_formula_use <- xtrue_formula
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
xglm <- glm(xtrue_formula_use , data=dfx[ xndx_train, ], family=binomial(link="logit"))
yhat <- predict(xglm, newdata=dfx[ xndx_test, ], type="response")
mean( f_logit_cost(y=y_test, yhat=yhat) )
```