ROWL_sample_code

Shuying Zhu

Sample code of single-stage restricted outcome weighted learning (ROWL) method

```
library("kernlab")

BRITR  <- function(Y,Trt,X,PrTX,Tng,Sng,tau,kernelChoice,bigC, delta=0.2,iter.Max=20, epsilon=1e-6){

  # process the data
  res  <- lm(Y~X)$residuals;
  Ys  <- abs(res); #Y^*
  As  <- sign(res)*Trt; #A^*

  n  <- length(Y);

  # set initial values
  temp.beta0  <- rnorm(1, mean=0, sd=0.1);
  temp.beta  <-  rnorm(n, mean=0, sd=0.1);

  W1  <- cbind(diag(0,n), diag(-1,n), diag(Sng) );

  K  <- kernelMatrix(kernelChoice, X);

  ubC  <- 1e3;

  bvec  <- c(-bigC*rep(1,n)/n, 0);
  rvec  <- c(ubC*rep(1,n), 1e-8);
  lvec  <- rep(0, 3*n);
  uvec  <- c(bigC*Ys/PrTX, rep(ubC,2*n));

  for(iter in 1:iter.Max){
    delta1  <- c(rep(1,n), delta*rep(1,n), delta*(Tng-tau));
    temp.I  <-  ((temp.beta0+kernelMult(kernelChoice, X, z=temp.beta)) > 0);
    S_I <- Sng*temp.I;
    S_I <- as.vector(S_I);
    H  <- cbind(diag(As), diag(-1,n), diag(S_I));
    W2 <- c(As, rep(-1,n), S_I);
    W  <- rbind(W1, W2);

    # To make sure the H matrix has a full rank to increase numerical stable.
    H.ipop  <- t(H)%*%K%*%H+diag(1e-1,nrow=dim(H)[2])

    fit <- ipop(-delta1,H.ipop,A=W,b=bvec,l=lvec,u=uvec,r=rvec);

    e.beta<- H%*%primal(fit)
```

```
    fx <- kernelMult(kernelChoice, X, z=e.beta);

    testbeta0 <- seq(-max(fx)-1, max(fx)+10, 0.1)
    beta0j <- -max(fx)-1
    tempval <- rep(sum(Ys/PrTX*pmax(1-As*beta0j-As*fx,0)), length(testbeta0))
    tempval1 = rep(NA, length(testbeta0))
    cnt = rep(length(Y), length(testbeta0))

    for (j in 1:length(testbeta0)){
      beta0j <- testbeta0[j]
      temp = Tng + Sng*((beta0j+fx)>0)
      if (sum(temp>tau) <1){
        tempval[j] <- sum(Ys/PrTX*pmin(pmax(1-As*beta0j-As*fx,0), 1))
      }
    }
    e.beta0 <- max(testbeta0[which(tempval==min(tempval))])
    # print (sum((e.beta-temp.beta)^2))
    if (sum((e.beta-temp.beta)^2)<epsilon){
      break
    }
    temp.beta  <- e.beta;
    temp.beta0  <- e.beta0;
  }
  # #####check conditions#####
  # sum(primal(fit)>0)
  # sum(primal(fit)<uvec)
  # sum(W%*%primal(fit)>bvec)
  # sum(W%*%primal(fit)<(bvec+rvec))

  if(iter >= iter.Max){
    print("Max iteration step was reached!");
  }

  if(class(kernelChoice)[1]=="vanillakernel"){
    return(c(e.beta0,t(X)%*%e.beta));
  } else {
    return(c(e.beta0,e.beta));
  }

}
```