homework

LZH-XX

12/5/2020

##1. The optimization target and their dual gap and dual residual

Lasso

$\bar{g}_i^* (u_i) = \max_{\alpha_i : |\alpha_i| \leq B} u_i \alpha_i - \lambda |\alpha_i| = B [|u_i| - \lambda]_+$

```
positive <- function(x){
  if(x >= 0){
    return(x)
  }
  return(0)
}

lasso.w_func <- function(alpha,lambda,A,...){
  return(2*t(A)%*%(A %*% alpha - y))
}

lasso.subgrad <- function(alpha,lambda,A,demension = 1,...){
  i <- demension
  n <- length(alpha)
  # First calculate the subgrad of |alpha|, sub1
  if (alpha[i] > 0 ){
    sub1 <- alpha[i]/norm(alpha,type = "O")
  }
  else if(alpha[i] < 0){
    sub1 <- alpha[i]/norm(alpha,type = "O")
  }
  else {
    seed <- runif(1,-1,1)
    sub1 <- lambda*seed * alpha[i]/norm(alpha,type = "O")
  }

  # Then calculate the grad of ||A\alpha -y||^2
  term <- vector(length =n)
  for (s in 1:n){
    term <- term + alpha[s]*A[s]
  }

  sub2 <- t(A[,i]) %*% term + t(term) %*% A[,i]
          - t(y) %*% A[,i] - t(A[,i])%*% y
```

```
  subgrad = sub1 +sub2
  return(subgrad)
}

lasso.gap <- function(alpha,lambda = 0.5,B,w,A,demension = 1){
  # A is a matrix
  i <- demension
  return(B*positive(t(A[,i])%*% w- lambda)+lambda*abs(alpha[i])+
           alpha[i]*t(A[,i])%*% w)
}

lasso.dualres <- function(alpha,lambda,w,A,demension = i){
  # first calculate the subgrad of g_i^*
  i <- demension
  flag <-  0
  eps <- 1e-5
  input <- -t(A[,i])%*%w
  if (input > eps && input <= B){
    g_sub <- lambda *input
  }
  else if(input < -eps && input >= -B){
    g_sub <- -lambda *input
  }
  else if(abs(input) <B){
    g_sub_right <- lambda * abs(input)
    g_sub_left  <- -lambda * abs(input)
    flag <- 1
    #g_sub is an interval
  }
  else {
    g_sub <- Inf
  }

  # When flag ==0 ,subgrad is grad, otherwise is a interval
  if (flag == 0){
      return (abs(alpha[i])-g_sub)
  }
  #  subgrad is a interval
  if (abs(alpha[i]) <= g_sub_right){
    return (0)
  }
  else {
    return (min(abs(alpha[i]-g_sub_right),abs(alpha[i]-g_sub_left) ))
    # Return the closet distance from the interval
  }


}
```

Hinge-Loss SVM

$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \mathcal{O}_A(\boldsymbol{\alpha}) := \frac{1}{n}\sum_{i=1}^n \varphi_i^*\left(-\alpha_i\right) + \frac{\lambda}{2}\left\|\frac{1}{\lambda n}\sum_{i=1}^n \alpha_i \boldsymbol{a}_i\right\|_2^2$

```r
svm.loss <- function(alpha,lambda){
  n <- length(alpha)

}

svm.gap <- function(alpha,lambda,B,w,A,y,demension = 1){
  # A is a matrix
  i <- demension
  n <- length(alpha)
  g_conj <- function(input,y){
    return(input*y/n)
  }
  g <- function(input,y){
    # Not certain
    # alpha_i * y_i in [0,1]
    return(input * y)
  }
  return(g_conj(-t(A[,i],y[i])%*% w)+ g(alpha[i],y[i])+alpha[i]* t(A[,i]) %*%w)
}
```

##2.Adaptive Sampling -based CD

###2.1 Gap-wise

```r
sample.gapwise <- function(alpha,loss.gap,...){
  n <- length(alpha)
  p <- numeric(length = n)

  for (i in 1:n){
    p[i] <- loss.gap(alpha = alpha,demension = i)
  }
  psum <- sum(p)
  p[i] <- p[i]/psum
  return(p)
}
```

###2.2 Adaptive

```r
sample.ada <- function(alpha,loss.dualres,...){
  n <- length(alpha)
  p <- numeric(length = n)
  second_term <- numeric(length = n)
  sigma <- 0.5
  eps <- 1e-5
  for (i in 1:n){
    m <- n
    k <- abs(loss.dualres(alpha,demension = i))
    if( k < eps){
      p[i] <- 0
      m <- m-1
      break
    }
    second_term[i] <- k*norm(A[,i],type ="F")
```

```
  }

  second_term <- second_term/sum(second_term)

  for (i in 1:n){
    if (p[i] > 0){
      p[i] <- sigma/m + second_term*(1-sigma)
      }
  }
  return(p)
}
```

###2.3 Uniform

```
sample.uniform <- function(alpha,...){
  n <- length(alpha)
  p <- numeric(length = n)
  for (i in 1:n){
    p[i] <- 1/n
  }
  return(p)
}
```

###2.4 Importance Sampling

```
sample.imp <- function(alpha,A,...){
  n <- length(alpha)
  p <- numeric(length = n)
  for (i in 1:n){
    p[i] <- norm(A[,i],type = "F")
  }
  p = p/sum(p)
  return(p)
}
```

##3.Coordinate descdent

```
CD <- function(alpha_0 =0,max_iter =1000,sample,loss,loss.subgrad,w_func,A,y)
  {
  n <- length(alpha_0)
  # A,y are given data
  record.gap <- numeric(length = max_iter)
  alpha <- alpha_0
  w <- w_func(alpha)
  iter <- 0
  while(iter < max_iter){
  i <- sample(alpha = alpha,w =w,A=A,y=y,...)
  # direction is a sample prob vector like alpha with only one none zero dim
  direction = vector(length = n)
  direction[i] = 1

  update <- direction * loss.subgrad(alpha =alpha,dimension =i ,A =A,y=y,...)
```

```
  alpha <- alpha - update
  w <- w_func(alpha,...)

  }
}
```

##4.Exmperiment

Collect data

example

5.Plot