ADMM lasso

Su Chen

October 25, 2016

ADMM lasso algorithm

```
ADMM_lasso = function(x, y, beta0, step, rho, lambda, iter_max=100000, precision=1e-10)
{
  p = length(beta0)
  beta = matrix(0, nrow = iter_max, ncol = p)
  beta[1,] = beta0
  nll_track_lasso = rep(0, iter_max)
  nll_track_lasso[1] = nll_lasso(beta0, x, y, lambda)
  delta = 1
  iter = 1
  u = rep(0, p)
  z = rep(0, p)
  M = solve( t(x) %*% x + diag(rho, p) )
  N = t(x) %*% y

  while ( (delta >= precision) && (iter < iter_max) )
  {
    beta[iter+1,] = M %*% (N + rho*(z - u))
    z_update = soft_threshold(beta[iter+1,]+u, lambda=lambda/rho)
    u_update = u + beta[iter+1,] - z_update
    nll_track_lasso[iter+1] = nll_lasso(beta[iter+1,], x, y, lambda)
    delta = abs(nll_track_lasso[iter+1] - nll_track_lasso[iter])/nll_track_lasso[iter]
    iter = iter + 1
    z = z_update
    u = u_update
  }

  return (list(iter, nll_track_lasso, beta))
}
```

run and compare beta with glmnet output

```
diabetesX = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment5/diabetesX.csv
diabetesY = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment5/diabetesY.csv
X = as.matrix(diabetesX)
Y = as.matrix(diabetesY)
Xs = scale(X)
Ys = scale(Y)
beta0 = rep(0, ncol(Xs))
n = nrow(Xs)

### use cv.glmnet to choose optimal lambda ###
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

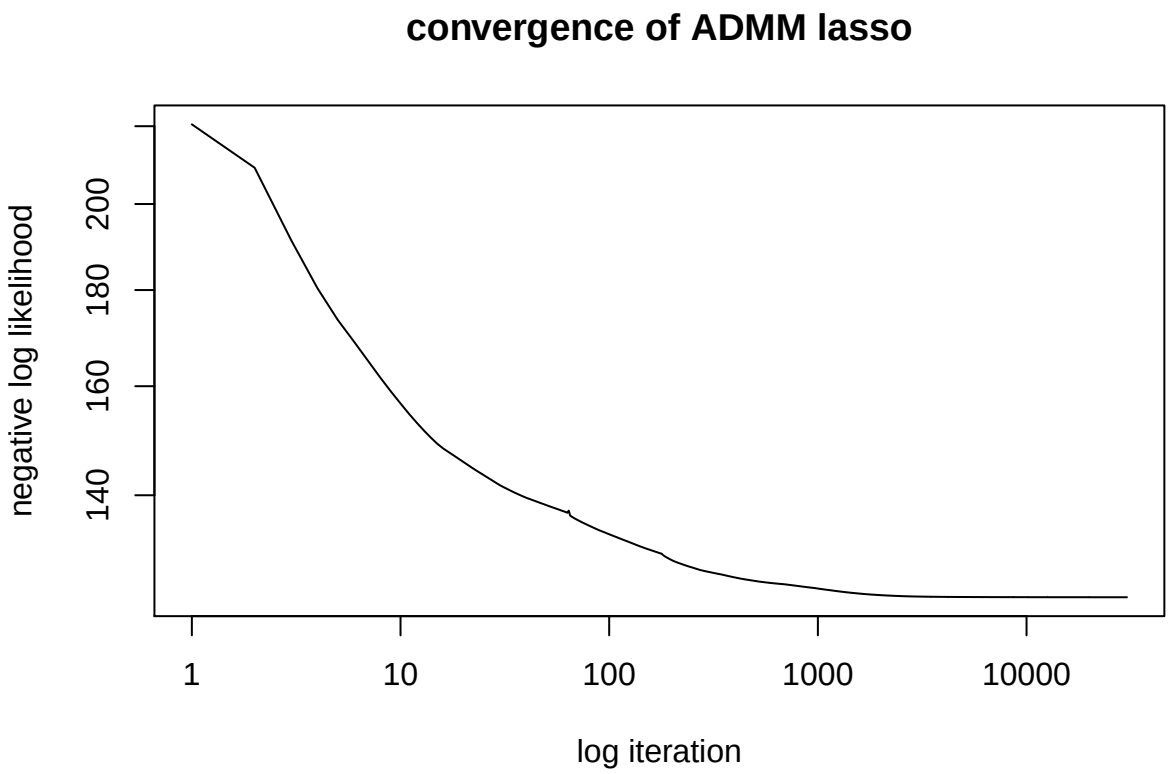
```
cv.fit = cv.glmnet(x = Xs, y = Ys)
cv.lambda = cv.fit$lambda.min
fit.cv.lambda = glmnet(x = Xs, y = Ys, lambda = cv.lambda)
glmnet_beta = fit.cv.lambda$beta
```

run three algrithms with optimal lambda scaled to number of data

```
result = prox_grad(x=Xs, y=Ys, beta0, step=1e-5, lambda=cv.lambda*n, iter_max=100000, precision=1e-10)
plot_iter = result[[1]]
result_nll_lasso = result[[2]][1:plot_iter]
result_beta = result[[3]][plot_iter,]


result_acc = acc_prox_grad(x=Xs, y=Ys, beta0, step=1e-5, s=2, lambda=cv.lambda*n, iter_max=100000, preci
plot_iter_acc = result_acc[[1]]
result_nll_lasso_acc = result_acc[[2]][1:plot_iter_acc]
result_beta_acc = result_acc[[3]][plot_iter_acc,]


result_ADMM = ADMM_lasso(x=Xs, y=Ys, beta0, rho=1,
                         lambda=cv.lambda*n, iter_max=100000, precision=1e-10)
plot_iter_ADMM = result_ADMM[[1]]
result_nll_lasso_ADMM = result_ADMM[[2]][1:plot_iter_ADMM]
result_beta_ADMM = result_ADMM[[3]][plot_iter_ADMM,]
plot(x = 1:plot_iter_ADMM, y = result_nll_lasso_ADMM, type = "l",
     xlab = "log iteration", ylab = "negative log likelihood",
     log = "xy", main = "convergence of ADMM lasso")
```

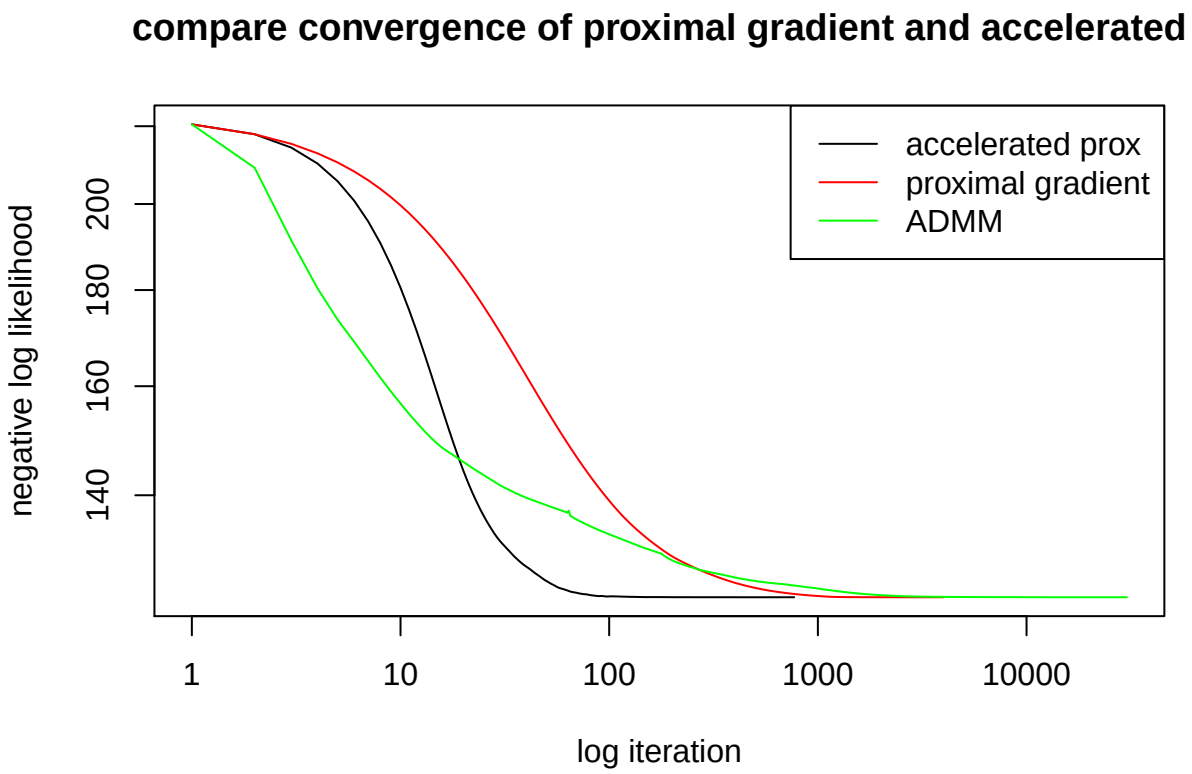


plot three nll to compare speed of convergence

```
plot_iter_comp = max(plot_iter, plot_iter_acc,plot_iter_ADMM)
result_nll_lasso_comp = result[[2]][1:plot_iter_comp]
result_nll_lasso_acc_comp = result_acc[[2]][1:plot_iter_comp]
result_nll_lasso_ADMM = result_ADMM[[2]][1:plot_iter_comp]
plot(x = 1:plot_iter_comp, y = result_nll_lasso_acc_comp, type = "l",
     xlab = "log iteration", ylab = "negative log likelihood",
     log = "xy", main = "compare convergence of proximal gradient and accelerated")
```

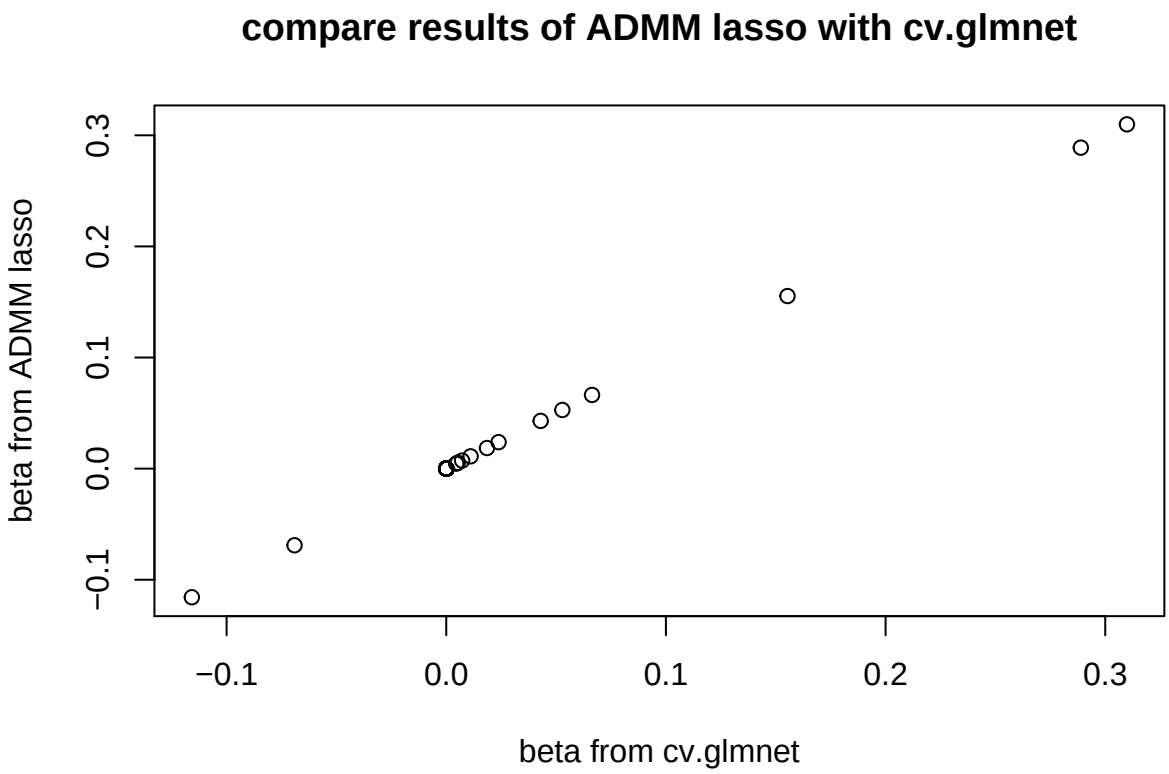
```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 29508 y values <= 0
## omitted from logarithmic plot
```

```
lines(x = 1:plot_iter_comp, y = result_nll_lasso_comp, col = "red")
lines(x = 1:plot_iter_comp, y = result_nll_lasso_ADMM, col = "green")
legend("topright", c("accelerated prox","proximal gradient","ADMM"),
       col = c("black", "red","green"), lty = c(1,1,1))
```



plot betas compare to glmnet output

```
plot(x = glmnet_beta, y = result_beta_ADMM, xlab = "beta from cv.glmnet",
     ylab = "beta from ADMM lasso",
     main = "compare results of ADMM lasso with cv.glmnet")
```


compare results of ADMM lasso with cv.glmnet
beta from ADMM lasso
−0.1
0.0
0.1
0.2
0.3
beta from cv.glmnet
−0.1
0.0
0.1
0.2
0.3