Peer Review evaluation 2

Reviewer: Giorgio Paulon
Reviewee: Mingzhang Yin

Preprocessing:

You had two really nice ideas:

- scale the X matrix without centring it in order to preserve sparsity;
- change every sparse matrix to a row oriented format in R *(X = as(X, "RsparseMatrix"))* instead of transposing the matrices.

Coding advice:

The code was working well and it was easy to read because we used a similar scheme. Just one remark that is not likely to drastically improve your code:

- You can pass objects to functions by reference. This will avoid a copy of all the big matrices every time a function is called (most notably, in the main function). If you analyse the performances of your code, you will notice that around 7 out of 15 seconds are spent just to copy the object in memory for C++. What you could do, instead, is writing a function caller in C++ that reads in the data from disk and then call the sgd function with referenced objects.

  List **sgdtest**(VectorXd Ytrain, MapMatd Xtrain, VectorXd Ytest, MapMatd Xtest,
  VectorXd beta, double lambda, double masterStepSize)

  and you should just write it as

  List **sgdtest**(VectorXd& Ytrain, MapMatd& Xtrain, VectorXd& Ytest, MapMatd& Xtest,
  VectorXd beta, double lambda, double masterStepSize)

Apart from this remark, your code is well written and respects every good programming rule!

Algorithm review:

You implemented the lazy update with the approximation we talked about in class. Using the exact formula for the updating scheme is not likely to change the predictive accuracy, which is already pretty good.

Choice of lambda:

Instead of choosing a value for the penalty term by hand, you could use cross-validation. I run a simple cross-validation scheme in R to choose the best lambda parameter:

```
N <- length(Ytrain)
n.folds = 5
idx <- sample(rep(1:n.folds, length.out = N))
lambda.vec <- c(0, 1E-10, 1E-9, 1E-8, 1E-7, 1E-6, 1E-5, 1E-4, 1E-3)
prec <- matrix(NA, nrow = n.folds, ncol = length(lambda.vec))
for (j in 1:length(lambda.vec)){
  for (i in 1:n.folds){
```

Peer Review evaluation 2

Reviewer: Giorgio Paulon
Reviewee: Mingzhang Yin

```
  # Select the test set indexes
  test <- idx == i
  # Run the algorithm
  ada <-
sgdtest(Ytrain[!test],Xtrain[!test,],Ytrain[test],Xtrain[test,],beta,lambda=lambda.vec[j],masterStep
Size=0.8)
  prec[i,j] <- ada$accuracy
  k <- k+1
  cat("Iteration: ", k)
 }
}
```

The result is:

$\lambda = 0$	$\lambda = 10^{-6}$	$\lambda = 10^{-5}$	$\lambda = 10^{-4}$	$\lambda = 10^{-3}$	$\lambda = 10^{-2}$	$\lambda = 10^{-1}$
0.9880057	0.9736554	0.9711431	0.9610852	0.8901395	0.7092270	0.4862591

The best accuracy is achieved for the model with no penalty, which is the analogous that I obtained with my version of the algorithm.