droplasso_vignette

Beyrem

4/24/2018

Introduction

Droplasso is a package that fits a generalized linear model via penalized maximum likelihood. The penalization term is a mixture of Lasso () and dropout penalties. As for now It fits least square and logistic regression models. A variety of predictions can be made from the fitted models. This vignette describes the usage of Droplasso in R.

Given training dataset x with labeles y, droplasso solves the following optimization problem

$$\underset{w\in\mathbb{R}^d}{min}\left(R_{dropLasso}(x,y,w)\right) = \underset{w\in\mathbb{R}^d}{min}\underset{\delta_1,\ldots,\delta_n\sim B(p)}{\mathbb{E}}?\left(\frac{1}{n}\sum_{i=1}^{n}L(y_i,w^\top(\delta_{\mathbf{i}}\odot x_{i,}))+\lambda.\left\|w\right\|_1\right) \tag{1}$$

where L is the loss or the negative log likelihood for an observation. The Lasso penalty is controlled by λ and the dropout penalty is controlled by p. The Droplasso becomes Lasso when $p = 1$ and dropout when $\lambda = 0$

Algorithm

our algorithm to solve problem () is based on a stochastic proximal gradient descent.

Installation

Quick Start

The purpose of this section is to give users a general sense of the package. We also compare to the glmnet package.

```
library(droplasso)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-16
```

Simulation : convergence and results

You can also embed plots, for example:

Now we try on the paper simulations and we evaluate performance

```
library(mvtnorm)


generate_count_data <- function(n,d,d1,lambda,p) {
```

```
  d2<-d-d1
  mu <- rep(0,d)
  Sigma <- matrix(1, nrow=d, ncol=d)  + diag(d)
  rawvars <- rmvnorm(n,mean=mu, sigma=Sigma)
  pvars <- pnorm(rawvars)
  ztrain  <- qpois(pvars, lambda)
  w <-c(rep(0.05,d1),rep(0,d2)) # sparse model

  z <- ztrain %*%  w
  pr <- 1/(1+exp(-z))         # pass through an inv-logit function
  y <- rbinom(n,1,pr)      # NOISY bernoulli response variable

  #bern <- qbinom(pvars, 1,p)
  #xtrain=ztrain * bern
xtrain <- sapply(1:d,function(i) ztrain[,i] * rbinom(n,1,p))
  return(cbind(xtrain,y))
}

train=generate_count_data(100,100,10,1,1)
xtrain=train[,1:(ncol(train)-1)]
y=train[,ncol(train)]



test=generate_count_data(10000,100,10,1,1)
xtest=test[,1:(ncol(test)-1)]
ytest=test[,ncol(test)]
```

```
library(ROCR)
```

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##     lowess
```

```
xtrain=scale(xtrain,center=F)
xtest=scale(xtest,center=F)
d1=10

# Main loop (done in parallel on several cores)

alpha=sapply(1:10,function(i)  (i-1)/9)
accuracy <- function(ind1,ind2)
{     n=nrow(xtrain)
    d=ncol(xtrain)

# Elasticnet
m_glm <- glmnet(xtrain,y , family="binomial" , nlambda=10 , intercept=F, alpha=alpha[ind2] ,standardize

ypred <- (xtest) %*% m_glm$beta
pred <- prediction(ypred[,ind1], ytest)
```

```r
auc_el <- performance(pred, "auc")@y.values[[1]]
a_el_p=length(which(abs(m_glm$beta[1:d1,ind1])>0))/d1
a_el_n=length(which(m_glm$beta[(d1+1):d,ind1]==0))/(d-d1)


# droplasso

m <-  droplasso(as.matrix(xtrain),y,family="binomial",lambda=m_glm$lambda[ind1]*alpha[ind2],
                keep_prob=1/(1+m_glm$lambda[ind1]*(1-alpha[ind2])/2),n_passes = 10)

ypred <- (xtest) %*% m
pred <- prediction(ypred[,1],ytest)
auc_dl <- performance(pred, "auc")@y.values[[1]]
a_dl_p=length(which(m[1:d1]>0))/d1
a_dl_n=length(which(m[(d1+1):d]==0))/(d-d1)


acc=c(auc_el,auc_dl,a_el_p,a_dl_p,a_el_n,a_dl_n)
return(acc)
}

seq=c(1:10)
ind_list=list()
for (i in 1:10)
    ind_list=append(ind_list,mapply(c,i,seq,SIMPLIFY = F))

total_acc <-sapply(ind_list, function(i) accuracy(i[1],i[2]))


max(total_acc[1,])
```

```
## [1] 0.6108854
```

```r
max(total_acc[2,])
```

```
## [1] 0.6086022
```